VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Generátor výstupních sestav

Report generator

2010 Bc. Adam Bartoníček

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 7.5.2010 …….

Podpis

Poděkování

Rád bych poděkoval vedoucí diplomové práce paní Ing. Emílii Šeptákové za odbornou pomoc a konzultaci při vytváření této práce.
Dále bych chtěl poděkovat mému kolegovi Ing. Jaromíru Sitkovi za rady při vytváření práce, pomoc při jejím testování a odborné korektury textu.

Abstrakt

Tato diplomová práce popisuje vývoj Generátoru výstupních sestav. Je to systém, který bude sloužit k vytváření výstupních sestav v podobě PHP skriptu z již existujících databází na databázových serverech, ke kterým se bude aplikace připojovat. V úvodní kapitole je popsáno zadání informačního systému. Poté následuje kapitola, která se věnuje popisu použitých technologií. Další kapitola rozebírá analýzu systému. Analýza je důležitá pro návrh implementace, který ji zpřesňuje a upravuje, aby bylo možné systém implementovat v konkrétním jazyku. V návrhu implementace je také popsáno zabezpečení aplikace. Implementace v programovacím jazyce C# jako ASP.NET webová aplikace s využitím technologie ODBC pro připojení k různým databázovým serverům, je popsána hned v následující kapitole. V této kapitole je popsán i grafický návrh aplikace, instalace systému a také ukázky některých funkcí. Následuje kapitola věnovaná testování systému. Závěrečná část obsahuje shrnutí zkušeností s návrhem a implementací celého systému.

Klíčová slova

ASP.NET, C#, databáze, datová analýza, datový slovník, DFD diagram, E-R diagram, funkční analýza, implementace, indexová analýza, informační systém, kontextový diagram, Microsoft SQL server, minispecifikace, MySQL server, ODBC, Oracle, PHP, stavová analýza, testovací scénář, typ sestavy, výstupní sestava

Abstract

This graduation thesis describes the development of a summary reports generator. It is a system which will be instrumental to generating of summary reports in a PHP script from databases in the databases servers to which it will be connected to. There is a setting of an information system described in the chapter I. Then follows a chapter describing applied technologies. The next chapter analyses the system analysis. That chapter is important for the implementing part because it makes it easier and faster. The implementation in the C# programming language with applying of the ODBC technology for connection to various databases servers is described in the consecutive chapter. There will be defined the application graphic design in that chapter. Then follows a chapter dedicated to the system testing. The concluding part contains the practice summary with the whole system definition and implementation.

Key words

ASP.NET, C#, database, data analysis, data dictionary, DFD diagram, E-R diagram, functional analysis, implementation, index analysis, information system, context diagram, minispecification, MySQL, server, Microsoft SQL server, ODBC, Oracle, PHP, status analysis, testing scenario, type of report, summary report

Seznam použitých symbolů a zkratek

.NET	Soubor technologií tvořících platformu pro vývoj aplikací pro web a windows
AJAX	„*Asynchronous JavaScript and XML*“. Technologie vývoje interaktivních webových aplikací, které mění svůj obsah bez nutnosti jejich znovunačítání
API	„*Application Programing Interface*“. Aplikační programové rozhraní
ASP.NET	Je to součást .NET pro tvorbu webových aplikací a služeb
COM	„*Component Object Model*“. Standart pro interprocesní komunikaci a dynamické vytváření objektů v programovacích jazycích
CSS	„*Cascading Style Sheets*“. Jazyk pro popis způsobu zobrazování HTML stránek
DFD diagram	„*Data Flow Diagram*“. Diagram datových toků. Slouží k modelování funkcí systému
DLL	„*Dynamic-link library*“. Dynamicky připojitelné knihovny
E-R diagram	„*Entity-Relationship diagram*“. Entitně-relační diagram, zobrazuje vztahy mezi evidovanými typy entit.
GPL	„*General Public Licence*“. Je to volná bezplatná softwarová licence
HTML	„*HyperText Markup Language*“. Jazyk pro vytváření internetových stránek
HTTP	„*HyperText Transefer Protocol*“. Internetový protokol používaný pro přenos informací
ID	Zkratka pro identifikaci. Zde jednoznačně identifikuje záznam v rámci typu entity v databázi.
IIS	„*Internet Information Services*“. Webový server nacházející se ve Windows
MSSQL	Microsoft SQL Server. Databázový server vytvořený firmou Microsoft
NULL	Tato hodnota znamená „nic“, bez hodnoty, prázdná hodnota
ODBC	„*Open Database Conectivity*“. Softwarové API pro jednotný přístup k databázovým serverům
PHP	„*Hypertext PreProcesor*“. Skriptovací programovací jazyk
RAM	Operační paměť počítače
SP	„*Service Pack*“. Opravný balíček pro softwarový produkt
SQL	„*Structured Query Language*“. Strukturovaný dotazovací jazyk používaný pro práci s daty v relačních databázích.
SSL	„*Secure Sockets Layer*“. Protokol pro zabezpečenou komunikaci
URL	„*Uniform Resource Locator*“. Je to řetězec znaků sloužící pro specifikaci umístění zdrojů informací v rámci internetu
XHTML	„*eXtensible HyperText Markup Language*“. Je to modifikace jazyka HTML tak, aby odpovídal podmínkám tvorby XML dokumentů()

Obsah

Seznam tabulek

Seznam obrázků

Seznam příkladů

1 Úvod

Tato diplomová práce se zabývá vývojem Generátoru výstupních sestav. Výsledná webová aplikace slouží k vytváření výstupních sestav z již existujících databází v podobě PHP skriptu. V první kapitole je specifikováno zadání práce, které bylo vytvořeno po konzultacích se zadavatelem, paní Ing. Emílií Šeptákovou. Součástí zadání práce je specifikace aktérů systému, dále vstupů do systému a výstupů systému. Hlavní částí zadání je ovšem specifikace funkcí vyvíjeného systému.

Další částí práce je popis použitých technologií, kde je popsáno, o jaký typ aplikace se jedná, kterých technologií bude využívat a jak jsou tyto technologie mezi sebou propojeny. Dále jsou zde popsány jednotlivé technologie, jejich možnosti, vlastnosti a historie. Do použitých technologií spadá programovací jazyk, ve kterém byla aplikace vyvíjena, dále databázové servery, ke kterým se aplikace bude umět připojit a také rozhraní pomocí kterého aplikace s těmito databázovými servery bude komunikovat. Ke všem technologiím jsou vloženy odkazy na literaturu, ze kterých může čtenář čerpat doplňující informace.

Následující kapitola se týká analýzy vyvíjené webové aplikace. Tato kapitola je na celé práci nejdůležitější, protože obsahuje analýzu typů sestav, ale také proto, že je potřebná pro následující části práce. Z tohoto důvodu je tato kapitola nejrozsáhlejší. Analýza typů sestav rozděluje výstupní sestavy na čtyři základní typy, které poté důkladně rozebírá. U každého typu sestavy jsou uvedeny jeho možnosti, vlastnosti a také je zde uvedeno několik příkladů konkrétní výstupní sestavy. Kromě již zmiňované analýzy typů sestav je v této části i datová, funkční a indexová analýza. Datová analýza rozebírá návrh databázové části celého projektu, to znamená typů entit, jejich atributů a jejich vztahů. Funkční analýza se zabývá návrhem funkcí systému, jejich propojením a také tím, které funkce používají které typy entit pro uchovávání údajů. Indexová analýza popisuje vytvoření indexů v databázi pro urychlení práce s ní.

Kapitola Návrh implementace popisuje návrh implementace vzhledem ke zvolenému programovacímu jazyku a zvolenému databázovému serveru. Je zde popsáno zabezpečení aplikace pomocí autentizace a autorizace uživatelů a jejich dělení do rolí. Také se zde nachází upravená datová analýza, kde došlo k přidání nových typů entit, a změnily se datové typy atributů. Stavová analýza se nachází v této kapitole, protože popisuje stavy objektu uživatel, které vznikly díky použití zabezpečení aplikace.

Část nazvaná Implementace rozebírá implementaci webové aplikace. Jsou zde uvedeny hardwarové a softwarové požadavky jak pro serverovou, tak pro klientskou část. Dále obsahuje soupis prostředků použitých pro vývoj aplikace, popis grafického návrhu a ukázky některých funkcí programu.

Následující kapitola Testování obsahuje ukázkové testovací scénáře, které byly použity při testování Generátoru výstupních sestav. Závěrečná kapitola obsahuje shrnutí vývoje celé aplikace.

2 Specifikace zadání

Zadavatelem práce je Ing. Emílie Šeptáková. Potřebuje vytvořit webovou aplikaci, která bude umožňovat, aby se uživatel - programátor, který je registrovaný v aplikaci, prostřednictvím této aplikace mohl připojit k libovolnému databázovému serveru, ke kterému má přístupové údaje, aby mohl vytvořit výstupní sestavu a případně tuto sestavu uložit pro pozdější použití. Databázovým serverem může být Microsoft SQL server, nebo MySQL server, nebo Oracle. Vytváření sestav by mělo být realizováno přehledným průvodcem, ve kterém uživatel zvolí typ sestavy, zdroj dat a jednotlivé prvky sestavy. Výstup sestavy by měl být ve spustitelném skriptu PHP. Po spuštění tohoto skriptu se provede připojení k databázi, vyhledání a zobrazení údajů sestavy. Uživateli by mělo být umožněno ukládat navázaná spojení s databázovými servery pro pozdější použití, a také ověřovat, zda lze se zadanými přístupovými údaji k databázovým serverům navázat spojení. Také by měla být k této aplikaci vypracovaná programátorská a uživatelská příručka. Na výsledné aplikaci se bude provádět testování.

2.1 Kdo bude se systémem pracovat

Systém bude víceuživatelský. Tito uživatelé budou rozděleni do dvou skupin. Na administrátory a běžné uživatele.

Běžní uživatelé budou moci vytvářet a ověřovat spojení s databázovými servery. Tato spojení budou moci ukládat a případně upravovat nebo mazat. Také budou moci vytvářet výstupní sestavy přes jednoduchého průvodce. Tyto sestavy budu moci ukládat, upravovat a mazat. Uživatelé si budou moci změnit přístupové heslo do aplikace, nebo si nechat zaslat emailem nově vygenerované heslo pro přihlášení v případě zapomenutí toho původního. Aplikace by měla být schopna si zapamatovat uživatelovo přihlášení, aby se uživatel nemusel do aplikace pokaždé přihlašovat. Všechny důležité změny v systému by měly být provedeny až po potvrzení uživatelem.

Administrátoři budou mít přístup ke všem funkcím informačního systému. Budou používat stejné funkce jako běžní uživatelé. Navíc k tomu mají možnost mazat spojení i sestavy od všech uživatelů v systému. Upravovat budou pouze své sestavy a své spojení. Jako hlavní ovšem je pro ně funkce vytváření nových uživatelů, jejich editace a mazání ze systému. Také budou moci všechny uživatele přiřazovat do skupin nebo jim skupinu měnit. Administrátoři mají možnost uživatele odblokovat, pokud je systém zablokuje z důvodu chybného zadání hesla 5x za sebou. Také budou moci uživatele blokovat a odblokovat pro přihlášení.

2.2 Vstupy do systému

Informace o spojení. Každé spojení je identifikováno jednoznačným ID, které přidělí systém při vytvoření, dále je u spojení evidováno, který uživatel jej vytvořil. Hlavní atributy spojení jsou ovšem IP adresa databázového serveru, se kterým se spojení navazuje a port, na kterém server běží. Dále je to uživatelské jméno a heslo pro přihlášení k tomuto serveru a nakonec typ databázového serveru.

O každém uživateli se eviduje jeho jednoznačné identifikační číslo, jeho přihlašovací jméno, heslo, email, na který se v případě zapomenutí dá odeslat nové heslo. Dále se eviduje otázka pro zaslání hesla a odpověď na tuto otázku. Systém potom eviduje, do které skupiny uživatel patří. Zda do administrátorů, nebo do uživatelů. Nakonec se ještě evidují systémové atributy, které uvidí pouze administrátor. To jsou například datum a čas vytvoření uživatele, datum a čas posledního přihlášení, jestli je uživatel připojený, zda je uživatel zablokovaný z důvodu zapomenutí hesla a zda je uživatel zablokovaný přímo administrátorem.

Sestava bude identifikována jednoznačným identifikačním číslem, bude obsahovat jméno uživatele, který ji vytvořil, dále spojení, které bude použito s databází, nadpis sestavy, název sestavy, počet zobrazených záznamů na stránku, datum vytvoření sestavy, typ sestavy, dále se budou evidovat všechny potřebné položky týkající se SQL příkazu, který bude reprezentovat sestavu. Nakonec se bude evidovat použitá šablona pro vytvoření PHP skriptu.

2.3 Výstupy ze systému

Uživatelé

- výpis jednotlivých spojení pro konkrétního uživatele
- výpis jednotlivých sestav pro konkrétního uživatele

Administrátoři

- výpis všech spojení v systému
- výpis všech sestav v systému
- výpis všech uživatelů registrovaných v systému

Obě skupiny

- email uživateli s nově vygenerovaným heslem v případě jeho zapomenutí
- PHP skript s vygenerovanou výstupní sestavou
- informativní a potvrzovací dialogy

2.4 Funkce systému

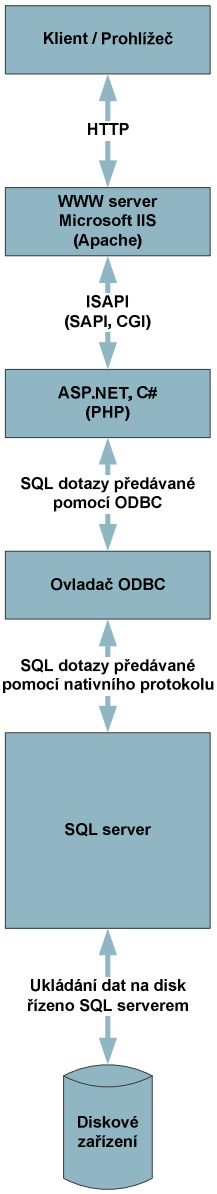
Událost	Reakce	Aktér (skupina)
Test spojení	Ověř údaje zadané uživatelem na validnost vstupu. Pokus se navázat spojení s databázovým serverem a podej hlášení o stavu spojení.	uživatel, administrátor
Vytvoření spojení	Proveď funkci Test spojení, a pokud je spojení v pořádku, tak zapiš nové spojení do databáze a informuj o tom uživatele.	uživatel, administrátor
Přehled spojení	Pro typ role uživatelé zobraz všechna jeho spojení s možností spojení vymazat, nebo editovat. Pro typ role administrátoři zobraz všechna spojení od všech uživatelů. Zobraz tlačítka pro vymazání spojení a editaci, která bude možná pouze u spojení vytvořených pro aktuálního uživatele. U spojení od jiných uživatelů bude tlačítko pro editaci deaktivované. Přehled bude stránkován po 10 spojeních.	uživatel, administrátor
Editace spojení	Vyber z databáze všechny údaje týkající se vybraného spojení a zobraz je uživateli ve formuláři pod přehledem spojení. Dané spojení v přehledu označ. Uživatel provede změnu údajů, poté proveď funkci Test spojení a zapiš aktualizované údaje zpět do databáze.	uživatel, administrátor
Vymazání spojení	Zobraz uživateli potvrzovací dialog o smazání spojení. Pokud uživatel chce opravdu smazat spojení, tak jej smaž z databáze, jinak neudělej nic.	uživatel, administrátor
Vytvoření sestavy	Zobraz uživateli průvodce, který umožní vytvořit sestavu. Uživatel vybere, nebo zadá potřebné údaje jako spojení, název databáze, nadpis, výběr dat atd. Nakonec ulož sestavu do databáze a nabídni uživateli stažení PHP skriptu s výstupní sestavou.	uživatel, administrátor
Přehled sestav	Pro typ role uživatelé, zobraz všechny jejich sestavy s možností sestavy vymazat, nebo editovat. Pro typ role administrátoři zobraz všechny sestavy od všech uživatelů. Zobraz tlačítka pro vymazání a editaci, která bude možná pouze u sestav vytvořených pro aktuálního uživatele. U jiných uživatelů bude tlačítko pro editaci deaktivované. Přehled bude stránkován po 10 spojeních.	uživatel, administrátor
Editace sestavy	Vyber z databáze všechny údaje týkající se konkrétní sestavy, načti je do průvodce a potom tohoto průvodce zobraz uživateli. Uživatel může změnit hodnoty. Poté tyto aktualizované hodnoty přepiš v databázi.	uživatel, administrátor

Událost	Reakce	Aktér (skupina)
Vymazání sestavy	Zobraz uživateli potvrzovací dialog o smazání sestavy. Pokud uživatel chce opravdu smazat sestavu, tak ji smaž z databáze, jinak neudělej nic.	uživatel, administrátor
Vytvoření uživatele	Zobraz administrátorovi dialog pro zadání informací o uživateli. Zkontroluj platnost vyplněných údajů. Např. správnost emailové adresy, její jedinečnost, jedinečnost uživatelského jména a sílu hesla. Poté zobraz výběr role uživatele. Nakonec ulož informace do databáze a informuj správce o výsledku.	administrátor
Přehled uživatelů	Administrátorovi zobraz všechny uživatele s tlačítky pro jejich editaci, nebo vymazání. Přehled bude stránkován po 10 uživatelích.	administrátor
Editace uživatele	Vyber z databáze všechny údaje týkající se vybraného uživatele a zobraz je administrátorovi ve formuláři pod přehledem uživatelů. Daného uživatele v přehledu označ. Administrátor provede změnu údajů. Poté ulož změněné údaje do databáze.	administrátor
Vymazání uživatele	Zobraz administrátorovi potvrzovací dialog o smazání uživatele. Pokud administrátor chce opravdu smazat uživatele, tak jej smaž z databáze, jinak neudělej nic.	administrátor
Přihlášení do aplikace	Zobraz dialog pro vyplnění uživatelského jména a hesla a také možnost pro zapamatování přihlášení. Uživatel vyplní údaje. Proveď kontrolu, zda uživatel existuje, zda není zablokovaný a zda je heslo správné. Pokud je vše v pořádku, autentizuj jej a zobraz úvodní stránku. Jinak zobraz chybové hlášení.	uživatel, administrátor
Odhlášení z aplikace	Zobraz dialog pro potvrzení odhlášení. Pokud uživatel souhlasí, odhlas jej z aplikace a přesměruj jej na přihlašovací stránku. Jinak nedělej nic.	uživatel, administrátor
Změna hesla	Zobraz formulář pro zadání původního hesla a nového. Uživatel vyplní údaje. Poté proveď změnu hesla v databázi.	uživatel, administrátor
Obnova hesla	Zobraz dialog pro obnovu hesla. Uživatel vyplní uživatelské jméno. Poté zobraz kontrolní otázku. Uživatel vyplní odpověď. Proveď její kontrolu. Pokud je vše v pořádku, odešli na uživatelův email nově vygenerované heslo a také jej zapiš do databáze.	uživatel, administrátor

Tabulka 1: Funkce systému

3 Použité technologie

Výsledná aplikace bude naprogramována jako webová aplikace. Generátor sestav bude dostupný zadáním URL adresy, na které bude umístěn, do libovolného internetového prohlížeče. Tato webová aplikace bude naprogramována s využitím technologie ASP.NET v programovacím jazyce C#. Bude umožňovat propojení s databázovými servery MySQL, Microsoft SQL Server a Oracle. Pro spojení jazyka C# a PHP s databázovými servery se bude používat softwarové rozhraní ODBC. Na obrázku níže je uveden diagram, který zobrazuje, jakým způsobem jsou jednotlivé technologie mezi sebou propojeny a jakým způsobem je mezi nimi zabezpečena komunikace a předávání dat.

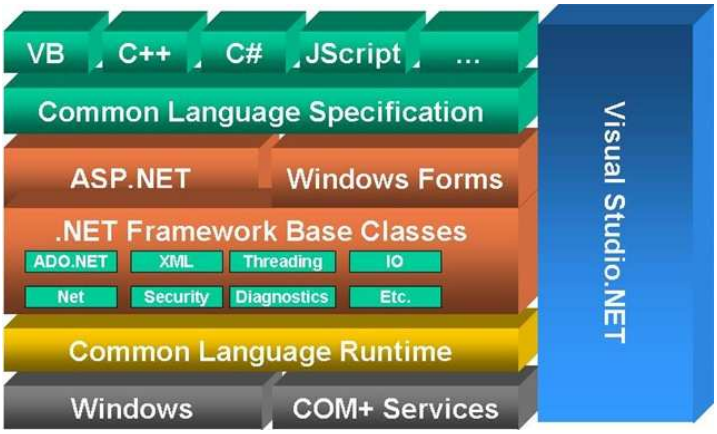


Obrázek 1: Propojení jednotlivých technologií

3.1 .NET

.NET je název pro soubor technologií, které tvoří platformu pro vývoj aplikací nejen pro web a operační systém Windows, ale i pro přenosná zařízení s operačním systémem Windows. Tento soubor technologií pochází od firmy Microsoft a byl poprvé představen v roce 2002, ale jeho vývoj trval již od 90. let minulého století. .NET poskytuje nástroje, které dovolují vývojářům vytvářet programy pro Windows mnohem rychleji a snadněji. Aplikace jsou potom mnohem bezpečnější, kvalitnější a dostupnější. Aby mohl uživatel spustit .NET aplikaci, musí mít nainstalován .NET Framework, který je oddělený od operačního systému.

3.1.1 Popis architektury



Obrázek 2: Architektura .NET

Architektura .NET se skládá ze čtyř hlavních částí:

Common Language Specification (CLS) – na obrázku zeleně. Je to obecná specifikace jazyků, která integruje kód a komponenty z více .NET programovacích jazyků. Aplikace v .NET může být napsána více programovacími jazyky bez zbytečné práce navíc. .NET zahrnuje nové objektově orientované programovací jazyky jako C#, Visual Basic, .NET, J# a řízené C++. Tyto jazyky se poté kompilují do CLS a mohou společně pracovat v jedné aplikaci.

Framework Class Library (FCL) – na obrázku oranžově. FCL je kolekce více než 7000 tříd a datových typů, které umožňují aplikacím .NET číst a zapisovat soubory, zpracovávat XML soubory, přistupovat k databázím, zobrazovat grafické uživatelské rozhraní, kreslit, používat webové služby, používat prvky zabezpečení a diagnostiky a mnoho dalších možností. FCL rozděluje mohutné Win32 API na více jednoduchých .NET objektů, které jsou použity v programovacích jazycích .NET.

Common Language Runtime (CLR) – na obrázku žlutě. CLR je spouštěcí část pro .NET aplikace a slouží jako rozhraní mezi nimi a operačním systémem. Poskytuje například následující služby:

- načítání a vykonávání kódu
- převod mezijazyka do strojového kódu pro konkrétní počítač
- oddělení procesů a paměti
- správa paměti a objektů
- správa zabezpečení
- zpracování výjimek
- rozhraní mezi řízeným kódem, COM objekty, a DLL knihovnami
- typovou kontrolu
- popisná data pro kód (meta data)
- ladění kódu

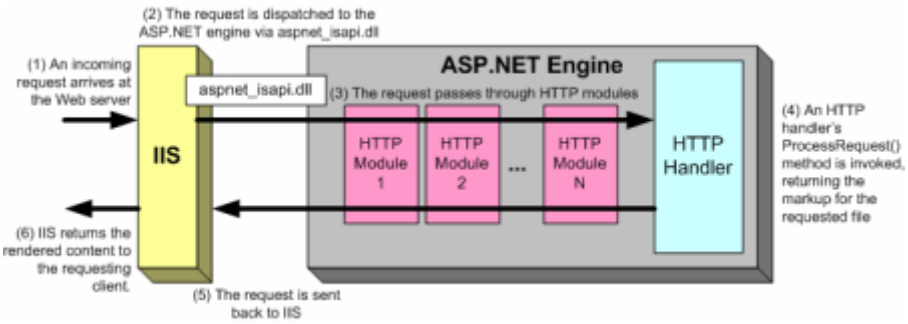
.NET Tools – na obrázku modře. Visual Studio .NET je software pro vývoj aplikací na této platformě. Je to integrované vývojové prostředí pro vývojáře, kterým poskytuje nástroje pro vytváření aplikací pro windows. Aplikace mohou být konzolové, desktopové a internetové.

Šedou barvou je na obrázku znázorněn operační systém.

Více informací o .NET lze nalézt ve dvou anglických zdrojích *wikipedia* [1] a *DevTopics* [2]

3.1.2 ASP.NET

ASP.NET je součástí .NET Framework Class Library a slouží pro vývoj webových aplikací a služeb. Je nástupcem technologie ASP (Active Server Pages). Od technologie ASP je ale hodně odlišná. Programátoři mohou pracovat v kterémkoliv jazyce z CLS (C#, Visual Basic, …), aplikace je mnohem rychlejší díky předkompilovaní do DLL souborů a také je tím odstraněno více chyb již při vývoji. Stránky jsou poskládány z objektů a ovládacích prvků, které jsou protějškem těch ve Windows. Proto lze používat prvky jako Button, Label atd. a těmto prvkům přiřazovat vlastnosti, odchytávat události apod. Webové ovládací prvky tvoří HTML kód, který tvoří část stránky, která se zobrazí klientovi. V novějších verzích je generovaný HTML kód validní. Lze definovat vlastní uživatelské ovládací prvky a použít je jako šablony, čímž se redukuje duplicitní kód. Také je možné ukládat do vyrovnávací paměti celou stránku, nebo jen její část. Ačkoliv webový protokol HTTP je bezstavový (neumí uchovat stav), událostmi řízené programování zachování stavu vyžaduje. ASP.NET řeší tento problém pomocí HTML kódu a JavaScriptu pomocí dvou základních technik:



Obrázek 3: Vykonání požadavku ASP.NET

ViewState – uchovává informace mezi opakovaným odesláním formuláře na server v zakódovaném tvaru ve skrytých formulářových polích. Používá k tomu HTML kód, tudíž není potřeba speciální podpora ani na straně serveru ani na straně klienta. Znamená to ale přenos většího množství informací.

Session state – uchovává všechny informace na straně serveru a předává (jako cookie, nebo v URL) pouze jednoznačný identifikátor. To zatěžuje více server, ale klade menší nároky na přenos dat. Tyto session jdou ukládat jako samostatný proces, nebo na SQL server. To umožňuje zachovat stav i po restartu serveru.

Podrobnosti o ASP.NET lze nalézt na *anglické wikipedii* [3]

3.2 PHP

izek 4: Logo PHP

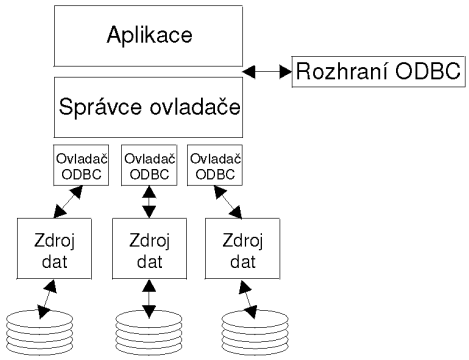
PHP – Hypertext preprocesor je univerzální skriptovací programovací jazyk určený především pro vývoj dynamických internetových stránek. Vytvořil jej Rasmus Lerdorf v roce 1995. Nyní PHP vyvíjí firma The PHP Group. Je k dispozici zdarma v rámci PHP licence, která je odlišná od licence GPL. Lze jej ale použít i k tvorbě konzolových aplikací. PHP skripty jsou vykonávány na straně serveru a k uživateli je přenášen až výsledek jejich činnosti. Vykonání PHP skriptu je možné i spuštěním PHP interpretu z příkazové řádky. Jazyk je inspirován například jazyky Pascal, C, Java, Perl. Jazyk je multiplatformní a lze jej tedy provozovat na různých operačních systémech např. na Microsoft Windows nebo na Linuxu. Podporuje mnoho knihoven pro různé účely a mnoho internetových protokolů. Podporuje mnoho databázových systémů nativně, nebo pomocí ODBC. Je podporován většinou hostingových služeb. V *české* [4] nebo *anglické* [5] literatuře můžete nalézt více informací. Na *domovských stránkách* [6] lze PHP stáhnout nebo prohlížet dokumentaci k němu.

3.3 ODBC

ODBC – Open DataBase Connectivity je standardizované softwarové API pro přístup k databázovým systémům. Vytvořila jej firma Microsoft v roce 1992. Díky ODBC mohou aplikace přistupovat k datům nezávisle na tom, na jakém databázovém serveru jsou data uložena i přesto, že každý systém řízení báze dat používá pro uchovávání dat jiný formát a jiné rozhraní pro přístup k nim. ODBC je multiplatformní. Lze jej provozovat nejen na operačních systémech od firmy Microsoft, ale třeba i na Unixových systémech nebo i na IBM. Dříve byl přístup k datům pomocí ODBC pomalý, ale postupnou optimalizací se rychlost přístupu zvýšila na úroveň nativních ovladačů.

Architektura ODBC se dělí na čtyři vrstvy.

- **Aplikace** – první nejvrchnější vrstva obsahuje samotnou aplikaci, ta provede v případě potřeby dat volání ODBC funkcí (SQL příkazem)
- **Správce ODBC ovladačů** – jehož úkolem je spojit aplikaci s příslušným ODBC ovladačem. Pokud aplikace potřebuje data, správce ovladačů vyhledá příslušný ovladač a nahraje jej do paměti, tímto může být prováděn přístup k více ovladačům současně.
- **ODBC ovladače** – provedou zpracování ODBC funkce, přeložení požadavku do SQL jazyka pro příslušný databázový systém a jeho následné odeslání.
- **Systém řízení báze dat** – provede zpracování SQL požadavku a výsledek vrátí zpět ovladači



tura ODBC

Princip a popis ODBC je popsán v následující literatuře *zde v češtině* [7] a *zde v angličtině* [8].

3.4 MySQL

MySQL je relační systém řízení báze dat, který vytvořila švédská firma MySQL AB v roce 1995. Dnes je majetkem firmy Oracle, která má také svůj systém řízení báze dat, viz níže. Je to multiplatformní systém. To znamená, že jej lze provozovat na různých operačních systémech. Například na Microsoft Windows nebo na Linuxu, ale i na jiných. Komunikace s MySQL probíhá pomocí jazyka SQL. Jedná se o dialekt tohoto jazyka s několika rozšířeními. Tento systém řízení báze dat je spravován pod dvěma typy licencí. Buď jako bezplatný volně šiřitelný (GPL), nebo jako komerční. Je to nejpoužívanější bezplatný systém jak v komerční sféře, tak i mezi amatéry vzhledem ke své výkonnosti, multiplatformnosti a také bezplatnosti. Z počátku byl systém optimalizován na co největší rychlost za cenu některých zjednodušení. Např. nepodporoval triggery, pohledy, uložené procedury a dokonce ani poddotazy. V novějších verzích již všechny tyto vlastnosti podporuje. Podrobnější informace lze nalézt v literatuře v *češtině* [9], nebo v *angličtině* [10]. Na *domovské stránce* [11] se můžete dočíst o všech vlastnostech nebo číst dokumentaci.

ek 6: Logo MySQL

3.5 Microsoft SQL server

:008

Microsoft SQL je relační databázový systém vyvinutý firmou Microsoft. První verze byla uvedena na trh v roce 1989. Kód pro tento systém pocházel z Sybase SQL serveru. Od roku 1994 je již Microsoft SQL server vyvíjen vlastním kódem. Tento systém je tak robustní a spolehlivý, že tvoří databázovou část důležitých firemních aplikací. Jeho nevýhoda je v tom, že je dostupný pouze pro operační systémy Microsoft Windows. Tím se od ostatních databázových systémů liší. Systém podporuje jazyk T-SQL, což je rozšíření jazyka SQL. Podporuje vnořené dotazy, triggery, uložené procedury a další pokročilé funkce SQL jazyka. Microsoft SQL server je dostupný v mnoha edicích zaměřených na různé uživatele. SQL server Enterprise edition je verze obsahující celou databázovou část a k ní spoustu programů pro její správu. Je to placená verze. Verze SQL server Express edition je sice zdarma, má však mnoho omezení. Jako podporu jen jednoho procesoru, 1GB RAM a 4GB místa na disku. Databáze se ukládá do jednoho souboru a neobsahuje programy pro správu. Verze SQL server Desktop engine je verze obsahující databázové jádro a je používaná ve Visual Studiu nebo Office Developeru. Je to opět hodně zjednodušená verze podobná Express edition. Programy pro správu jsou u tohoto relačního databázového systému velice užitečné. Podporují kromě běžných funkcí jako grafické rozhraní pro zadávání SQL příkazů, editaci uživatelů a práv a zálohování také funkce jako například frontu dat, replikační služby, služby pro dolování dat a jejich analýzu, fulltextové vyhledávání, generování zpráv. Více informací se můžete dozvědět na *tomto zdroji* [12], který je v angličtině, nebo na *domovských stránkách Microsoft SQL serveru* [13] v češtině.

3.6 Oracle Database

: Logo Oracle Corpo-

Oracle Database, jednodušeji Oracle, je moderní multiplatformní systém řízení báze dat s vysokým výkonem, škálovatelností a velmi pokročilými možnostmi zpracování dat. Systém byl vyvinut společností Software Deelopment Laboratories v roce 1977, která se později přejmenovala na Oracle Corporation. Tento databázový systém je v komerční sféře nejrozšířenějším. Je postaven na jazyku SQL s firemními rozšířeními podporujícími hierarchické dotazy, objektové databáze a jazyk PL/SQL, který je imperativní – rozšiřuje možnosti SQL o triggery, uložené procedury, funkce, programové balíky a další. Podporuje také nasazení v uzlových sítích, kde umožňuje sdílet zdroje z různých uzlů (počítačů). Tím naroste výkon mnohonásobně. Databázový systém Oracle se distribuuje ve více verzích. Verze Express edition je zdarma, ale má mnoho omezení. Je omezena na jeden procesor, pouze na operační systémy Windows a Linux, maximálně 4GB uživatelských dat. Verze Enterprise a Standard jsou placené a jsou prakticky bez omezení. Mnoho informací, ale pouze v angličtině lze nalézt v těchto zdrojích: *wikipedia* [14] a *domovská stránka systému* [15].

4 Analýza

Analýza je nejdůležitější částí vývoje informačního systému. Jejím cílem je rozebrat výsledný systém a definovat potřebné části pro implementační část. Špatná nebo málo důkladná analýza má za následek obtížnou nebo opakovanou implementaci nebo dokonce systém neodpovídající požadavkům zadavatele.

V průběhu analýzy jsem využil znalostí z předmětů Teorie zpracování dat, Databázové a informační systémy a Informační systémy a datové sklady. K těmto předmětům je k dispozici literatura. Pro *Teorii zpracování dat* [16], pro *Databázové a informační systémy* [17] a pro *Informační systémy a datové sklady* [18].

4.1 Analýza typů sestav

Data pro výstupní sestavy se ze systémů řízení báze dat získávají pomocí jazyka SQL a konkrétně pomocí příkazu SELECT. Více informací o tomto příkazu lze nalézt např. v knize SQL Hotová řešení v kapitole Výběr údajů [19].

4.1.1 Příkaz SELECT

Příkaz SELECT slouží pro získávání dat z databázových tabulek nebo z pohledů. Jeho možnosti jsou velmi široké. Dokáže vybrat všechny záznamy z tabulky nebo provést selekci (výběr množiny záznamů), nebo projekci (výběr množiny atributů). Také umožňuje změnit pořadí sloupců ve výpisu, změnit jejich názvy, vymazat duplicitní záznamy. Dále je schopný výsledky různě třídit, seskupovat, provádět agregační funkce apod. Jednou z jeho nejsložitějších funkcí je ovšem výběr dat z více tabulek. Toto lze provést mnoha způsoby. Vzhledem k tomu, že úplná syntaxe příkazu SELECT je velmi složitá, uvádím zde jeho zjednodušenou variantu:

```
SELECT [*], [seznam_položek_výstupní_sestavy]
      FROM jméno_tabulky
      WHERE podmínka_výběru
      GROUP BY položky
      HAVING podmínka_agregace
      ORDER BY seznam_položek [ASC][DESC];
```

V tomto příkaze jsou povinná pouze klíčová slova SELECT a FROM, zbytek je volitelný.

Příkaz začíná slovem **SELECT,** za kterým je předpis pro výběr sloupců. Je možno vybrat všechny sloupce tabulky pomocí * nebo vyjmenovat požadované sloupce v určitém pořadí. Také je zde možné přejmenovat sloupec pro výpis nebo použít agregační funkci.

Za klauzulí **FROM** se specifikuje zdroj požadovaných informací. Tím může být tabulka, více tabulek nebo pohled.

Za klíčovým slovem **WHERE** se určují podmínky, které specifikují podmnožinu vybíraných údajů nebo zde lze stanovit podmínku pro výběr údajů z více tabulek, nebo spojovací podmínku pro spojení více tabulek.

GROUP BY je klauzule, která umožňuje provedení seskupení údajů podle různých pravidel, která jsou v ní definována.

HAVING umožňuje definovat podmínky nad výslednými agregovanými (seskupenými) záznamy.

Klauzulí **ORDER BY** se určuje způsob setřídění údajů.

Z důvodu kompatibility mezi všemi databázovými serveru ukončuji příkaz středníkem. Takto jej přijmou všechny databázové servery.

Ukázkové výstupy budou vytvořeny pomocí MySQL serveru verze 5. Pokud některé příkazy nebude podporovat, bude uvedena obecná varianta a pod ní její náhrada v MySQL serveru.

Výstupní sestavy lze rozdělit na několik typů, podle toho, z jakého počtu tabulek se skládají, také podle vzhledu výsledné sestavy a použitých částí příkazu SELECT.

4.1.2 Výběr dat z jedné tabulky

Tento typ výstupní sestavy je nejjednodušší. Příkaz SELECT pracuje pouze s jednou databázovou tabulkou. Pro tento typ sestavy je možno definovat spoustu výstupních podmínek, třídících podmínek apod. Na tomto typu sestavy uvedu příklady pro definování těchto podmínek i s ukázkovými SQL příkazy a jejich výstupy v tabulkách na cvičných datech. Tato data mají 10 záznamů a 8 atributů. Jedná se o data ze skladu zboží v prodejně.

4.1.2.1 Výběr všech údajů z tabulky

Tento výběr se provede příkazem SELECT s použitím * jako zástupného znaku za všechny atributy konkrétní tabulky. Příkaz vrátí jako výsledek všechny záznamy z tabulky a všechny atributy záznamů v tabulce seřazené tak, jak jsou uloženy v databázi.

Příklad 1: Výběr všech údajů z tabulky sklad

SQL Příkaz:

```
SELECT *
    FROM sklad;
```

Popis SQL příkazu:

Tento příkaz vypíše všechny záznamy (konkrétně 10) v tabulce sklad se všemi jejich atributy seřazené tak, jak jsou uložené v databázi. Zde konkrétně seřazeny podle atributu id vzestupně.

Výsledek SQL příkazu:

id	ean	nazev	mnozstvi	cena	zal_cas	trvanlivost	trv_dnu
0	8594012320123	Sýrové tyčinky 90g	90	9.00	2008-01-09 15:44:09	1	10
1	8592794011529	Flint kokosová tyčinka bílá poleva 50g	20	6.00	2008-01-09 15:57:01	1	10
2	8593894809726	Miňonky smetanové 50g	0	2.00	2008-01-09 16:02:31	1	10
3	85911512	Banány v čokoládě 50g	0	8.00	2008-01-09 16:03:29	1	10
4	85906679	Koko v čokoládě 40g	3	7.00	2008-01-09 16:04:22	1	10
5	8594014033236	Mini bageta se šunkou 155g	5	19.00	2008-01-09 16:19:06	0	0
6	8594014871029	Chipsy solené ARO 80g	67	8.00	2008-01-09 16:22:45	1	10
7	8584004030000	Mila polomáčené 50g	13	8.00	2008-01-09 16:24:32	1	10
8	50985098	Halls extra strong 33,5g	9	16.00	2008-01-09 16:28:16	1	10
9	50985081	Halls original 33,5g	3	16.00	2008-01-09 16:31:25	1	10

Tabulka 2: Výběr všech údajů z tabulky sklad

4.1.2.2 Výběr pouze některých atributů tabulky (projekce)

Tento výběr se provede příkazem SELECT s definováním, které sloupce se budou vypisovat, ze které tabulky a v jakém pořadí. Příkaz vrátí všechny záznamy v tabulce a u nich jen některé atributy.

Příklad 2: Výběr některých atributů z tabulky sklad

SQL Příkaz:

```
SELECT nazev, ean, cena, mnozstvi
    FROM sklad;
```

Popis SQL příkazu:

Příkaz vypíše všechny záznamy (10) z tabulky sklad a k nim jen atributy nazev, ean, cena a mnozstvi. Atributy jsou vypsány v pořadí, v jakém jsou zapsány v příkazu. V tomto případě nazev, ean, cena, množství. Pořadí příkazů v databázové tabulce je jiné.

Výsledek SQL příkazu:

nazev	ean	cena	mnozstvi
Sýrové tyčinky 90g	8594012320123	9.00	90
Flint kokosová tyčinka bílá poleva 50g	8592794011529	6.00	20
Miňonky smetanové 50g	8593894809726	2.00	0
Banány v čokoládě 50g	85911512	8.00	0
Koko v čokoládě 40g	85906679	7.00	3
Mini bageta se šunkou 155g	8594014033236	19.00	5
Chipsy solené ARO 80g	8594014871029	8.00	67
Mila polomáčené 50g	8584004030000	8.00	13
Halls extra strong 33,5g	50985098	16.00	9
Halls original 33,5g	50985081	16.00	3

Tabulka 3: Výběr některých atributů z tabulky sklad

4.1.2.3 Výběr údajů podle podmínky

Tento výběr se provede příkazem SELECT s definováním podmínky výběru za klauzuli WHERE. Definice atributů není potřeba. Může se použít *, nebo vypsat, které atributy mají být ve výpisu. Příkaz vrátí všechny záznamy odpovídající podmínce. Podmínek se může definovat i více než jedna. Podmínky jsou potom odděleny logickými spojkami (AND, OR). Podmínka umožňuje porovnávat hodnotu atributu nebo hodnotu funkce, či hodnotu definovanou uživatelem s jinou hodnotou atributu, funkce, či s jinou hodnotou definovanou uživatelem pomocí komparačních operátorů (=, <>, >, <, >=, >=).

Příklad 3: Výběr některých atributů z tabulky sklad podle dvou podmínek

SQL Příkaz:

```
SELECT nazev, ean, cena
     FROM sklad
     WHERE trvanlivost=1
          AND mnozstvi>0;
```

Popis SQL příkazu:

Příkaz vypíše pouze záznamy, které splní podmínku, že mají trvanlivost (mohou se zkazit) a jejich množství je větší než nulové z tabulky sklad. Jinými slovy vypíše zboží, které je na skladě, a může se zkazit. K těmto záznamům vypíše pouze atributy nazev, ean a cena. Záznamy jsou seřazeny tak, jako jsou v tabulce, to je podle id.

Výsledek SQL příkazu:

nazev	ean	cena
Sýrové tyčinky 90g	8594012320123	9.00
Flint kokosová tyčinka bílá poleva 50g	8592794011529	6.00
Koko v čokoládě 40g	85906679	7.00
Chipsy solené ARO 80g	8594014871029	8.00
Mila polomáčené 50g	8584004030000	8.00
Halls extra strong 33,5g	50985098	16.00
Halls original 33,5g	50985081	16.00

Tabulka 4: Výběr některých atributů z tabulky sklad podle dvou podmínek

4.1.2.4 Výběr údajů se seřazením

Tento výběr provede příkaz SELECT, pro který platí, že se může použít výběr sloupců, nebo také nemusí, nemusí se definovat ani podmínka za klauzulí WHERE. Seřazení se definuje za klauzulí ORDER BY.

Příklad 4: Výběr některých atributů z tabulky sklad s podmínkou a se seřazením podle dvou podmínek

SQL Příkaz:

```
SELECT nazev, ean, cena
    FROM sklad
    WHERE cena>=8
    ORDER BY cena ASC, ean DESC;
```

Popis SQL příkazu:

Příkaz vypíše záznamy z tabulky sklad a k nim atributy nazev, ean a cena, kde cena je větší nebo rovna 8Kč a výsledek seřadí nejprve podle ceny vzestupně a poté, pokud je cena stejná, podle čárového kódu sestupně.

Výsledek SQL příkazu:

nazev	ean	cena
Chipsy solené ARO 80g	8594014871029	8.00
Banány v čokoládě 50g	85911512	8.00
Mila polomáčené 50g	8584004030000	8.00
Sýrové tyčinky 90g	8594012320123	9.00
Halls extra strong 33,5g	50985098	16.00
Halls original 33,5g	50985081	16.00
Mini bageta se šunkou 155g	8594014033236	19.00

Tabulka 5: Výběr některých atributů z tabulky sklad s podmínkou a se seřazením podle dvou podmínek

4.1.2.5 Výběr seskupených údajů

Seskupování se definuje v příkazu SELECT za klauzulí GROUP BY. Provádí se podle hodnot určitých atributů. Pokud je potřeba seskupovat podle více atributů, napíše se za klauzuli GROUP BY seznam těchto atributů oddělených čárkami. Princip seskupování spočívá v tom, že když se objeví dva záznamy, jejichž hodnoty se v seskupovaných atributech shodují, databázový systém na ně bude nahlížet jako na jeden. Lze si to představit jako shluknutí několika řádků do jednoho. Pro tuto skupinu lze spočítat pomocí agregačních funkcí některé matematické operace.

Agregační funkce

COUNT (*) – je nejjednodušší agregační funkce. Slouží k získání počtu záznamů v rámci jedné skupiny agregovaných záznamů.

COUNT (atribut) – slouží k získání počtu nenulových záznamů daného atributu.

SUM (atribut) – funkce spočítá součet hodnot v uvedeném atributu v rámci shluknuté skupiny záznamů.

AVG (atribut) – funkce počítá aritmetický průměr z hodnot příslušného atributu shluklých záznamů.

MIN (atribut) – funkce spočítá minimum ze seskupených záznamů.

MAX (atribut) – funkce vrátí maximum ze seskupených záznamů.

Příklad 5: Výběr počtu vět v tabulce sklad

SQL Příkaz:

```
SELECT COUNT(*) AS pocet
    FROM sklad;
```

Popis SQL příkazu:

Příkaz je ukázkou použití nejjednodušší agregační funkce, která počítá pouze počet záznamů v tabulce Sklad. Sloupec COUNT (*) je přejmenován na počet. Není zde využito seskupování údajů klauzulí GROUP BY. Výsledkem tohoto příkazu je počet vět v databázové tabulce sklad.

Výsledek SQL příkazu:

pocet
10

Tabulka 6: Výběr počtu vět v tabulce sklad

Příklad 6: Výpis některých agregačních funkcí v tabulce sklad seskupené podle tvanlivosti

SQL Příkaz:

```
SELECT trvanlivost, COUNT(*) AS pocet, AVG(cena) AS prumernaCe-
na, SUM(mnozstvi) AS mnozstviCelkem, MAX(mnozstvi) AS maximumM-
nozstvi
     FROM sklad
     GROUP BY trvanlivost;
```

Popis SQL příkazu:

Příkaz vypíše hodnoty některých agregačních funkcí pro zvolené atributy v tabulce, která je seskupená podle hodnoty atributu trvanlivost. Trvanlivost má jen 2 hodnoty, proto má výstup jen 2 řádky. Je zde ukázáno přejmenování výsledných sloupců. Pro každou seskupenou hodnotu se počítá průměrná cena, maximální množství atd.

Výsledek SQL příkazu:

trvanlivost	pocet	prumernaCena	mnozstviCelkem	maximumMnozstvi
0	1	19.000000	5	5
1	9	8.888889	205	90

Tabulka 7: Výpis některých agregačních funkcí v tabulce sklad seskupené podle tvanlivosti

4.1.2.6 Výběr seskupených údajů s podmínkou pro agregaci

Agregační podmínka se definuje za klauzulí HAVING. Její vlastnosti jsou stejné jako pro podmínku definovanou za klauzulí WHERE. Princip agregační podmínky spočívá v tom, že tato podmínka se neaplikuje na původní záznamy, ale až na seskupené.

Příklad 7:Výběr některých agregačních funkcí z tabulky sklad seskupené podle trvanlivosti s podmínkou pro seskupení

SQL Příkaz:

```
SELECT trvanlivost, COUNT(*) AS pocet, AVG(cena) AS prumernaCe-
na, SUM(mnozstvi) AS mnozstviCelkem, MAX(mnozstvi) AS maximumM-
nozstvi
     FROM sklad
     GROUP BY trvanlivost
     HAVING COUNT(*)>5;
```

Popis SQL příkazu:

Příkaz vypíše hodnoty některých agregačních funkcí pro zvolené atributy v tabulce, která je seskupená podle hodnoty atributu trvanlivost. Trvanlivost má jen 2 hodnoty, proto má výstup jen 2 řádky. Je zde ukázáno přejmenování výsledných sloupců. Pro každou seskupenou hodnotu se počítá průměrná cena, maximální množství atd. V tomto konkrétním případě je vypsán jen jeden řádek (záznam) a to ten, který má počet záznamů ve skupině větší než 5.

Výsledek SQL příkazu:

trvanlivost	pocet	prumernaCena	mnozstviCelkem	maximumMnozstvi
1	9	8.888889	205	90

Tabulka 8: Výběr některých agregačních funkcí z tabulky sklad seskupené podle trvanlivosti s podmínkou pro seskupení

4.1.3 Výběr dat z více tabulek

Příkaz SELECT pracuje s více databázovými tabulkami. Pro tento příkaz je možno definovat výstupní podmínky, třídící podmínky apod., stejně jako u typu výstupní sestavy výběr dat z jedné tabulky. Více tabulek se musí v tomto příkazu spojit pomocí spojovacích podmínek. Pro definici těchto podmínek je možné použít klauzuli WHERE, nebo klauzuli JOIN ON a UNION. Tyto spojovací podmínky budou vysvětleny dále. Data pro ukázkové výstupy budou čtyři tabulky. Tabulka s firmami, kde je 5 záznamů, tabulka s výdejkami, kde jsou 4 výdejky pro jednu firmu a jedna výdejka pro třetí firmu. Dále je zde tabulka se zbožím, která obsahuje 18 záznamů. Z toho 2 nejsou na žádné výdejce. Nakonec tabulka se zaměstnanci, kterých je 5. Tento typ výstupní sestavy je složitější než předchozí typy dotazů.

4.1.3.1 Spojování tabulek pomocí klauzule WHERE

Tento druh spojování slouží například k tomu, že pokud z jedné tabulky je potřeba vypsat údaje, kde by bylo nějaké id odkazující se na druhou tabulku, a místo tohoto id je třeba vypsat jiný atribut z odkazované tabulky, použije se k tomu klauzule WHERE. Toto je ukázáno na příkladu 8.

Příklad 8: Výběr některých atributů z tabulek vydejky a firmy spojených pomocí id firmy

SQL Příkaz:

```
SELECT vydejky.id, vydejky.cislo, vydejky.cas, firmy.nazev
     FROM vydejky, firmy
     WHERE vydejky.id_firma = firmy.id;
```

Popis SQL příkazu:

Příkaz vypíše některé informace o výdejce a k těm zobrazí místo ID firmy z tabulky vydejky její název z tabulky firmy.

Výsledek SQL příkazu:

id	cislo	cas	nazev
11	MSG2090011	2009-06-24 08:23:39	MORAVIA SPORT GROUP spol. s r.o.
14	MSG2090012	2009-07-01 11:19:55	MORAVIA SPORT GROUP spol. s r.o.
15	MSG2090013	2009-07-09 07:41:14	MORAVIA SPORT GROUP spol. s r.o.
22	ANA2090017	2009-07-21 18:05:28	Nakladatelství ANAGRAM s.r.o.

Tabulka 9: Výběr některých atributů z tabulek vydejky a firmy spojených pomocí id firmy

Klauzule WHERE jde použít i v případě, že je potřeba spojit dvě tabulky, které spolu souvisí nepřímo, přes další tabulku. Tento typ je ukázán na příkladu 9.

Příklad 9: Spojení dvou tabulek pomocí třetí tabulky

SQL Příkaz:

```
SELECT firmy.nazev, zbozi.nazev
     FROM firmy, zbozi, vydejky
     WHERE vydejky.id_firma = firmy.id
          AND vydejky.id = zbozi.id_vydejky
          AND vydejky.id_firma=1;
```

Popis SQL příkazu:

Příkaz vypíše v tabulce všechno zboží, které se vydalo pro firmu s ID=1. U tohoto zboží vypíše pouze název firmy a název zboží. Toto zboží je zjištěno přes tabulku výdejky.

Výsledek SQL příkazu:

nazev	nazev
MORAVIA SPORT GROUP spol. s r.o.	TURMOIL
MORAVIA SPORT GROUP spol. s r.o.	OASIS
MORAVIA SPORT GROUP spol. s r.o.	VARIANT
MORAVIA SPORT GROUP spol. s r.o.	PONTE
MORAVIA SPORT GROUP spol. s r.o.	TRANSFER L
MORAVIA SPORT GROUP spol. s r.o.	TRANSFER XL
MORAVIA SPORT GROUP spol. s r.o.	RETRO 50 BLACK
MORAVIA SPORT GROUP spol. s r.o.	GETTER
MORAVIA SPORT GROUP spol. s r.o.	PROVIDER
MORAVIA SPORT GROUP spol. s r.o.	SHOPPER

Tabulka 10: Spojení dvou tabulek pomocí třetí tabulky

Zajímavá možnost může být spojení tabulky samé se sebou. Tohoto se využívá například, pokud tabulka obsahuje hierarchii nadřízený – podřízený, nebo pokud je třeba vytvořit rozpis „každý s každým“. Tento rozpis se využívá při různých turnajích. Ukázka vytvoření tohoto rozpisu je uvedena v následujícím příkladu.

Příklad 10: Rozpis pro fiktivní turnaj firem "každý s každým"

SQL Příkaz:

```
SELECT f1.nazev, f2.nazev
     FROM firmy f1, firmy f2
     WHERE f1.id < f2.id;
```

Popis SQL příkazu:

Příkaz vypíše ve dvou sloupcích rozpis pro fiktivní turnaj společností ve stylu každý s každým.

Výsledek SQL příkazu:

nazev	nazev
MORAVIA SPORT GROUP spol. s r.o.	PROMET LOGISTICS a.s.
MORAVIA SPORT GROUP spol. s r.o.	Nakladatelství ANAGRAM s.r.o.
PROMET LOGISTICS a.s.	Nakladatelství ANAGRAM s.r.o.
MORAVIA SPORT GROUP spol. s r.o.	LEARKA a.s.
PROMET LOGISTICS a.s.	LEARKA a.s.
Nakladatelství ANAGRAM s.r.o.	LEARKA a.s.
MORAVIA SPORT GROUP spol. s r.o.	BONATRANS a.s.
PROMET LOGISTICS a.s.	BONATRANS a.s.
Nakladatelství ANAGRAM s.r.o.	BONATRANS a.s.
LEARKA a.s.	BONATRANS a.s.

Tabulka 11: Rozpis pro fiktivní turnaj firem "každý s každým"

4.1.3.2 Spojování tabulek pomocí klauzule JOIN

Klauzule WHERE umožňuje provádět jen základní spojení. V praxi je ovšem potřeba provádět mnohem složitější spojení. Toho lze docílit klauzulí JOIN. Ta umožňuje provádět spojování, které má ve spojovacích atributech hodnoty NULL. Podle druhu spojení se tyto hodnoty berou v úvahu, nebo se ignorují a to jak v jedné z tabulek, tak případně v obou. Spojování tabulek pomocí JOIN lze rozdělit do čtyř skupin:

- Vnitřní spojení (INNER JOIN)
- Vnější spojení (OUTER JOIN), které může být levé (LEFT OUTER JOIN), nebo pravé (RIGHT OUTER JOIN)
- Křížové spojení (CROSS JOIN)
- Úplné spojení (FULL JOIN)

Vnitřní spojení (INNER JOIN)

Tabulka bude po spojení obsahovat jen ty záznamy, kde se hodnoty spojovacích atributů shodují. V tomto případě lze vytvořit spojení buď pomocí klauzule WHERE , nebo pomocí klauzule INNER JOIN. Ve spojené tabulce se neobjeví záznamy, kde je hodnota spojovacího atributu rovna NULL na jedné z tabulek nebo na obou.

Vnitřní spojení je vykonáváno na základě shody společných atributů. Výsledná tabulka bude obsahovat jen ty záznamy, ve kterých se hodnoty spojovacích atributů přesně shodují. Z tohoto spojení jsou vyloučeny záznamy, jejichž spojovací atributy přesně nekorespondují. Spojení má dvě strany. Levou a pravou. U vnitřního spojení nezáleží na volbě stran. Jen se změní pořadí výpisu sloupců při výpisu.

Syntaxe pro INNER JOIN je:

```
SELECT [*], [seznam_atributů]
      FROM prvni_tabulka
      INNER JOIN druha_tabulka
      ON podminka_spojeni;
```

Pří klad 11: Vnitřní spojení tabulek vydejky a firmy

SQL Příkaz:

```
SELECT *
      FROM vydejky v
      INNER JOIN firmy f
      ON v.id_firma = f.id;
```

Popis SQL příkazu:

Příkaz vypíše všechny atributy z tabulky výdejky a vedle nich všechny atributy z tabulky firma. Tímto se spojí tyto dvě tabulky v jednu. Vypíší se jen odpovídající záznamy. Protože v tabulce vydejky jsou výdejky pouze pro jednu firmu, je ve výpisu uvedena pouze tato firma.

Výsledek SQL příkazu:

id	cislo	cas	id_firma	id_zamestnanec	id	nazev	zal_cas
11	MSG2090011	2009-06-24 08:23:39	1	1	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
14	MSG2090012	2009-07-01 11:19:55	1	1	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
15	MSG2090013	2009-07-09 07:41:14	1	2	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
22	ANA2090017	2009-07-21 18:05:28	3	3	1	Nakladatelství ANAGRAM s.r.o.	2009-06-09 12:07:14

Tabulka 12: Vnitřní spojení tabulek vydejky a firmy

Pomocí vnitřního spojení lze spojovat dvě tabulky, které spolu přímo nesouvisí. Spojí se pomocí třetí tabulky.

Příklad 12: Spojení dvou tabulek pomocí třetí tabulky

SQL Příkaz:

```
SELECT f.nazev, z.jmeno
     FROM firmy f
     INNER JOIN vydejky v
     ON f.id=v.id_firma
     INNER JOIN zamestnanec z
     ON z.id=v.id_zamestnanec;
```

Popis SQL příkazu:

Příkaz vypíše název firmy a k ní zaměstnance, který tuto firmu obsluhoval přes výdejku.

Výsledek SQL příkazu:

nazev	jmeno
MORAVIA SPORT GROUP spol. s r.o.	Adam
MORAVIA SPORT GROUP spol. s r.o.	Adam
MORAVIA SPORT GROUP spol. s r.o.	Pavel
Nakladatelství ANAGRAM s.r.o.	Rostislav

Tabulka 13: Spojení dvou tabulek pomocí třetí tabulky

Vnější spojení z levé strany (LEFT OUTER JOIN)

U tohoto spojení se vypíší i řádky, které spojovací podmínku nesplňují. Spojení z levé strany znamená, že se budou přiřazovat záznamy z pravé tabulky k těm z levé. To znamená, že se vypíší i ty záznamy z levé tabulky, které v pravé tabulce žádný odpovídající (splňující spojovací podmínku) záznam nemají, ale z pravé tabulky se záznamy bez odpovídajícího (splňujícího spojovací podmínku) záznamu v levé tabulce neobjeví. Vnější spojení, ať už pravé, nebo levé se také používá k vyhledávání osamocených záznamů. Toho lze dosáhnout správným definováním podmínky spojení.

Syntaxe pro LEFT OUTER JOIN je:

```
SELECT [*], [seznam_atributů]
      FROM leva_tabulka
      LEFT OUTER JOIN prava_tabulka
      ON podminka_spojeni;
```

Příklad 13: Vnější levé spojení tabulek vydejky a firmy

SQL Příkaz:

```
SELECT *
      FROM vydejky v
      LEFT JOIN firmy f
      ON v.id_firma = f.id;
```

Popis SQL příkazu:

Příkaz spojí tabulky výdejky a firmy a zobrazí i ty záznamy, které kritérium nesplňují na levé straně. To znamená, že zobrazí všechny výdejky i ty, které nemají přiřazenu firmu.

Výsledek SQL příkazu:

id	cislo	cas	id_firma	id_zamest-nanec	id	nazev	zal_cas
11	MSG2090011	2009-06-24 08:23:39	1	1	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
14	MSG2090012	2009-07-01 11:19:55	1	1	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
15	MSG2090013	2009-07-09 07:41:14	1	2	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
22	ANA2090017	2009-07-21 18:05:28	3	3	1	Nakladatelství ANAGRAM s.r.o.	2009-06-09 12:07:14
106	ANA209006	2009-11-04 12:13:11	NULL	3	NULL	NULL	NULL

Tabulka 14: Vnější levé spojení tabulek vydejky a firmy

Vnější spojení z pravé strany (RIGHT OUTER JOIN)

U tohoto spojení se také vypíší řádky, které spojovací kritérium nesplňují. U spojení z pravé strany jsou přiřazovány záznamy z levé tabulky k těm z pravé. To znamená, že se vypíší i ty záznamy z pravé tabulky, které v levé tabulce žádný odpovídající záznam nemají, ale z levé tabulky se záznamy bez odpovídajícího záznamu v levé tabulce neobjeví.

Syntaxe pro RIGHT OUTER JOIN je:

```
SELECT [*],[seznam_atributů]
      FROM leva_tabulka
      RIGHT OUTER JOIN prava_tabulka
      ON podminka_spojeni;
```

Příklad 14: Vnější pravé spojení tabulek vydejky a firmy

SQL Příkaz:

```
SELECT *
      FROM vydejky v
      RIGHT JOIN firmy f
      ON v.id_firma = f.id;
```

Popis SQL příkazu:

Příkaz spojí tabulky výdejky a firmy a zobrazí i ty záznamy, které kritérium nesplňují na pravé straně. To znamená, že zobrazí všechny firmy i ty, které nemají ještě žádnou výdejku.

Výsledek SQL příkazu:

id	cislo	cas	id_firma	id_zamest-nanec	id	nazev	zal_cas
11	MSG2090011	2009-06-24 08:23:39	1	1	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
14	MSG2090012	2009-07-01 11:19:55	1	1	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
15	MSG2090013	2009-07-09 07:41:14	1	2	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
NULL	NULL	NULL	NULL	NULL	2	PROMET LOGIS-TICS a.s.	2009-05-21 14:00:45
22	ANA2090017	2009-07-21 18:05:28	3	3	1	Nakladatelství ANAGRAM s.r.o.	2009-06-09 12:07:14
106	ANA209006	2009-11-04 12:13:11	NULL	3	NULL	Nakladatelství ANAGRAM s.r.o.	2009-06-09 12:07:14
NULL	NULL	NULL	NULL	NULL	4	LEARKA a.s.	2009-11-23 08:14:53
NULL	NULL	NULL	NULL	NULL	5	BONATRANS a.s.	2010-02-23 11:33:25

Tabulka 15: Vnější pravé spojení tabulek vydejky a firmy

Úplné spojení (FULL JOIN)

U tohoto typu spojení se vypíší všechny řádky, které splňují spojovací kritérium a navíc i všechny záznamy z obou stran (ty, které kritérium nesplňují). Jinými slovy, provede se levé i pravé spojení zároveň. Tento druh spojení se nachází pouze v Microsoft SQL serveru nebo v Oracle. V MySQL se nenachází.

Pro MySQL se FULL JOIN nahrazuje spojením levého a pravého spojení. Toto je ukázáno na následujícím příkladu.

Syntaxe pro FULL JOIN je:

```
SELECT [*][seznam_atributů]
      FROM leva_tabulka
      FULL JOIN prava_tabulka
      ON podminka_spojeni;
```

Příklad 15: Úplné spojení tabulek firmy a vydejky

SQL Příkaz:

```
SELECT *
     FROM vydejky v
     FULL JOIN firmy f
     ON v.id_firma = f.id;
```

SQL Příkaz v MySQL:

```
SELECT *
     FROM vydejky
     LEFT JOIN firmy
     ON vydejky.id_firma = firmy.id
UNION ALL
SELECT *
     FROM vydejky
     RIGHT JOIN firmy
     ON vydejky.id_firma = firmy.id
     WHERE vydejky.id_firma IS NULL;
```

Na příkladě pro MySQL je vidět, že úplné spojení je sjednocení levého a pravého spojení.

Popis SQL příkazu:

Příkaz vrátí všechny záznamy odpovídající podmínce i všechny záznamy z obou stran, které podmínce neodpovídají. Provede se spojení tabulek výdejky a firmy.

Výsledek SQL příkazu:

id	cislo	cas	id_firma	id_zamest-nanec	id	nazev	zal_cas
11	MSG2090011	2009-06-24 08:23:39	1	1	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
14	MSG2090012	2009-07-01 11:19:55	1	1	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
15	MSG2090013	2009-07-09 07:41:14	1	2	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
22	ANA2090017	2009-07-21 18:05:28	3	3	1	Nakladatelství ANAGRAM s.r.o.	2009-06-09 12:07:14
106	ANA209006	2009-11-04 12:13:11	NULL	3	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	2	PROMET LOGIS-TICS a.s.	2009-05-21 14:00:45
NULL	NULL	NULL	NULL	NULL	4	LEARKA a.s.	2009-11-23 08:14:53
NULL	NULL	NULL	NULL	NULL	5	BONATRANS a.s.	2010-02-23 11:33:25

Tabulka 16: Úplné spojení tabulek firmy a vydejky

Křížové spojení (CROSS JOIN)

Křížové spojení vytvořené klauzulí CORSS JOIN vybere kartézský součin spojení. Tedy „každý s každým“. Toto se provede pouze tehdy, pokud není definována žádná spojovací podmínka. Toho se využívá například pro generování testovací množiny údajů. Křížové spojení lze aplikovat i na tabulku samu se sebou a tím vytvořit rozpis pro turnaj typu každý s každým. Pro tento případ se ale musí definovat ještě omezující podmínka, která bude ukázána na příkladu.

Syntaxe pro CROSS JOIN je:

```
SELECT [*][seznam_atributů]
     FROM prvni_tabulka
     CROSS JOIN druha_tabulka;
```

Příklad 16: Křížové spojení (kartézský součin) tabulek vydejky a firmy

SQL Příkaz:

```
SELECT *
     FROM vydejky v
     CROSS JOIN firmy f;
```

Popis SQL příkazu:

Příkaz provede kartézský součin (každý s každým) tabulek výdejky a firmy

Výsledek SQL příkazu:

id	cislo	cas	id_firma	id_zamest-nanec	id	nazev	zal_cas
11	MSG2090011	2009-06-24 08:23:39	1	1	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
14	MSG2090012	2009-07-01 11:19:55	1	1	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
15	MSG2090013	2009-07-09 07:41:14	1	2	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
22	ANA2090017	2009-07-21 18:05:28	3	3	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
106	ANA209006	2009-11-04 12:13:11	NULL	3	1	MORAVIA SPORT GROUP spol. s r.o.	2009-05-18 10:56:34
11	MSG2090011	2009-06-24 08:23:39	1	1	1	Nakladatelství ANAGRAM s.r.o.	2009-06-09 12:07:14
14	MSG2090012	2009-07-01 11:19:55	1	1	1	Nakladatelství ANAGRAM s.r.o.	2009-06-09 12:07:14
15	MSG2090013	2009-07-09 07:41:14	1	2	1	Nakladatelství ANAGRAM s.r.o.	2009-06-09 12:07:14
22	ANA2090017	2009-07-21 18:05:28	3	3	1	Nakladatelství ANAGRAM s.r.o.	2009-06-09 12:07:14
106	ANA209006	2009-11-04 12:13:11	NULL	3	1	Nakladatelství ANAGRAM s.r.o.	2009-06-09 12:07:14
11	MSG2090011	2009-06-24 08:23:39	1	1	2	PROMET LOGIS-TICS a.s.	2009-05-21 14:00:45
14	MSG2090012	2009-07-01 11:19:55	1	1	2	PROMET LOGIS-TICS a.s.	2009-05-21 14:00:45
15	MSG2090013	2009-07-09 07:41:14	1	2	2	PROMET LOGIS-TICS a.s.	2009-05-21 14:00:45
22	ANA2090017	2009-07-21 18:05:28	3	3	2	PROMET LOGIS-TICS a.s.	2009-05-21 14:00:45
106	ANA209006	2009-11-04 12:13:11	NULL	3	2	PROMET LOGIS-TICS a.s.	2009-05-21 14:00:45
11	MSG2090011	2009-06-24 08:23:39	1	1	4	LEARKA a.s.	2009-11-23 08:14:53
14	MSG2090012	2009-07-01 11:19:55	1	1	4	LEARKA a.s.	2009-11-23 08:14:53

id	cislo	cas	id_firma	id_zamest-nanec	id	nazev	zal_cas
15	MSG2090013	2009-07-09 07:41:14	1	2	4	LEARKA a.s.	2009-11-23 08:14:53
22	ANA2090017	2009-07-21 18:05:28	3	3	4	LEARKA a.s.	2009-11-23 08:14:53
106	ANA209006	2009-11-04 12:13:11	NULL	3	4	LEARKA a.s.	2009-11-23 08:14:53
11	MSG2090011	2009-06-24 08:23:39	1	1	5	BONATRANS a.s.	2010-02-23 11:33:25
14	MSG2090012	2009-07-01 11:19:55	1	1	5	BONATRANS a.s.	2010-02-23 11:33:25
15	MSG2090013	2009-07-09 07:41:14	1	2	5	BONATRANS a.s.	2010-02-23 11:33:25
22	ANA2090017	2009-07-21 18:05:28	3	3	5	BONATRANS a.s.	2010-02-23 11:33:25
106	ANA209006	2009-11-04 12:13:11	NULL	3	5	BONATRANS a.s.	2010-02-23 11:33:25

Tabulka 17: Křížové spojení (kartézský součin) tabulek vydejky a firmy

Příklad 17: Rozpis fiktivního turnaje firem pomocí křížového spojení

SQL Příkaz:

```
SELECT f1.nazev, f2.nazev
     FROM firmy f1
     CROSS JOIN firmy f2
     WHERE f1.nazev != f2.nazev
          AND f1.id < f2.id;
```

Popis SQL příkazu:

Příkaz provede úplné spojení tabulky samé se sebou. Tím by vznikl kartézský součin. Ten ovšem obsahuje duplicitní údaje a také údaje nesmyslné pro turnaj. Tyto se musí odstranit pomocí podmínky za klauzulí WHERE. Tímto vznikne rozpis turnaje.

Výsledek SQL příkazu:

nazev	nazev
MORAVIA SPORT GROUP spol. s r.o.	PROMET LOGISTICS a.s.
MORAVIA SPORT GROUP spol. s r.o.	Nakladatelství ANAGRAM s.r.o.
PROMET LOGISTICS a.s.	Nakladatelství ANAGRAM s.r.o.
MORAVIA SPORT GROUP spol. s r.o.	LEARKA a.s.
PROMET LOGISTICS a.s.	LEARKA a.s.
Nakladatelství ANAGRAM s.r.o.	LEARKA a.s.
MORAVIA SPORT GROUP spol. s r.o.	BONATRANS a.s.
PROMET LOGISTICS a.s.	BONATRANS a.s.
Nakladatelství ANAGRAM s.r.o.	BONATRANS a.s.
LEARKA a.s.	BONATRANS a.s.

Tabulka 18: Rozpis fiktivního turnaje firem pomocí křížového spojení

4.1.3.3 Sjednocení různorodých tabulek pomocí klauzule UNION

Někdy je třeba vypsat údaje ze dvou nehomogenních tabulek. Např. je jedna tabulka obsahující údaje o zákaznících (firmách) a druhá o pracovnících firmy. Obě obsahují údaj o jménu osoby, ale v jiném tvaru. Pokud je třeba vypsat jména všech zákazníků a k tomu i pracovníků firmy, použije se k tomu operátor UNION.

Syntaxe pro UNION je:

```
SELECT [*], [seznam_atributů]
      FROM prvni_tabulka
UNION
SELECT [*], [seznam_atributů]
      FROM druha_tabulka;
```

Příklad 18: Sjednocení atributu z tabulky firmy a atributu z tabulky zamestnanec

SQL Příkaz:

```
SELECT nazev
      FROM firmy
UNION
SELECT jmeno
      FROM zaměstnanec;
```

Popis SQL příkazu:

Příkaz udělá to, že sloučí dva atributy do jednoho. Vezme všechny názvy firem a pod ně vypíše jako další záznamy všechny jména zaměstnanců. Tento konkrétní příkaz nemá žádný praktický význam.

Výsledek SQL příkazu:

nazev
MORAVIA SPORT GROUP spol. s r.o.
PROMET LOGISTICS a.s.
Nakladatelství ANAGRAM s.r.o.
LEARKA a.s.
BONATRANS a.s.
Adam
Pavel
Rostislav
Vanda
Dita

Tabulka 19: Sjednocení atributu z tabulky firmy a atributu z tabulky zamestnanec

Operátor UNION lze použít i na rozčlenění položek výpisu z jedné tabulky na více typů.

Příklad 19: Sjednocení dvou atributů z jedné tabulky podle podmínky

SQL Příkaz:

```
SELECT jmeno
      FROM zaměstnanec
      WHERE id >3
UNION
SELECT heslo
      FROM zaměstnanec
      WHERE id >3;
```

Popis SQL příkazu:

Příkaz vypíše pod sebe hodnoty atributu jmeno a heslo z tabulky zaměstnanec, pokud je ID zaměstnance větší než 3. Tento příklad opět nemá žádný praktický význam. Slouží pouze pro demonstraci.

Výsledek SQL příkazu:

jmeno
Vanda
Dita
vvv
ddd

Tabulka 20: Sjednocení dvou atributů z jedné tabulky podle podmínky

4.1.4 Výběr dat 1:N

Tento typ výstupní sestavy není příliš složitý. Používá dva samostatné příkazy SELECT. Jeden příkaz je vnější a druhý je vnitřní. Vnitřní příkaz používá hodnoty z toho vnějšího. Nejčastěji je tento typ sestavy použit na faktuře nebo na prodejkách, výdejkách ze skladu apod. Sestava se skládá z hlavičky a z položek pro konkrétní hlavičku. U faktury je to např. hlavička odběratele, dodavatele, datum splatnosti, číslo faktury atd. Položky jsou v tomto případě prodané zboží na faktuře. Např. nějaké zboží, u kterého je vypsán jeho název, cena, množství, DPH, atd. Oba příkazy SELECT mohou obsahovat různé třídící podmínky, agregační funkce apod. Tyto byly všechny vysvětleny v kapitole věnované výběru dat z jedné tabulky, proto je zde již popisovat nebudu a uvedu zde jen příklad tohoto typu výstupní sestavy.

Příklad 20: Výpis výdejek a na nich příslušného zboží

SQL Příkaz 1:

```
SELECT *
    FROM výdejky;
```

Popis SQL příkazu 1:

Příkaz vypíše všechny výdejky v tabulce.

Výsledek SQL příkazu 1:

id	cislo	cas	id_firma	id_zamestnanec
11	MSG2090011	2009-06-24 08:23:39	1	1
14	MSG2090012	2009-07-01 11:19:55	1	1
15	MSG2090013	2009-07-09 07:41:14	1	2
22	ANA2090017	2009-07-21 18:05:28	3	3
106	ANA209006	2009-11-04 12:13:11	NULL	3

Tabulka 21: Výpis výdejek a na nich příslušného zboží příkaz 1

SQL Příkaz 2:

```
SELECT *
    FROM zbozi
    WHERE id_vydejky = 11;
```

Popis SQL příkazu 2:

Příkaz vypíše všechny atributy ke konkrétnímu zboží, které se nachází na konkrétní výdejce. Jinými slovy, vypíše všechno zboží na jedné výdejce. Nejprve vezme první výdejku s ID 11 a vypíše k ní všechno zboží. Potom se vezme výdejka s ID 14 a vypíše se zboží na ní. Tak se pokračuje dále

Výsledek SQL příkazu 2 (pro výdejku 11):

id	ean	nazev	kusu	id_vydejky
5845	8591130252336	TURMOIL	1	11
24004	8591130224333	OASIS	1	11
24008	8591130224371	VARIANT	2	11
24015	8591130226399	PONTE	1	11

Tabulka 22: Výpis výdejek a na nich příslušného zboží příkaz 2

Celková data:

id	cislo	cas	id_firma	id_zamestnanec
11	**MSG2090011**	**2009-06-24 08:23:39**	**1**	**1**
id	**ean**	**nazev**	**kusu**	**id_vydejky**
5845	8591130252336	TURMOIL	1	11
24004	8591130224333	OASIS	1	11
24008	8591130224371	VARIANT	2	11
24015	8591130226399	PONTE	1	11
14	**MSG2090012**	**2009-07-01 11:19:55**	**1**	**1**
id	**ean**	**nazev**	**kusu**	**id_vydejky**
33887	8591130321360	TRANSFER L	6	14
34247	5414709130657	TRANSFER L	6	14
15	**MSG2090013**	**2009-07-09 07:41:14**	**1**	**2**
id	**ean**	**nazev**	**kusu**	**id_vydejky**
29145	5414709111953	RETRO 50 BLACK	1	15
34457	5414709132736	GETTER	1	15
36692	5414709129750	PROVIDER	1	15
36693	5414709129897	SHOPPER	2	15
22	**ANA2090017**	**2009-07-21 18:05:28**	**3**	**3**
id	**ean**	**nazev**	**kusu**	**id_vydejky**
39163	5414709129316	ARCHER 55	1	22
106	**ANA209006**	**2009-11-04 12:13:11**	***NULL***	**3**
id	**ean**	**nazev**	**kusu**	**id_vydejky**
5046	8591130307197	RYE	21	106
5309	8591130307203	HAIDA	156	106
5485	8591130313600	SEOLON	11	106
5794	8591130307227	CHIAMA	23	106
33797	8591130300792	PANT	*1*	106

Tabulka 23: Výpis výdejek a na nich příslušného zboží - celková data

4.1.5 Výběr dat vazební tabulka

Tento typ sestavy je poněkud odlišný od ostatních. Jedná se o výpis, zda jsou dva typy entit ve vztahu M:N. Tento vztah je realizován vazební tabulkou. Vazební tabulka musí obsahovat jako atributy hodnoty primárních klíčů z jedné tabulky a hodnoty primárních klíčů z druhé tabulky. Sestava vypíše, zda konkrétní instance těchto typů entit jsou ve vazební tabulce, nebo ne. Samozřejmě lze pro tento typ sestavy definovat podmínky, řazení, filtry apod. jako u předchozích typů sestav

Výpis této sestavy bude tabulka, kde nadpisy sloupců budou hodnoty primárního klíče z jedné tabulky, nadpisy řádků budou hodnoty primárního klíče z druhé tabulky a obsah tabulky budou ANO/NE podle toho, zda konkrétní hodnoty klíčů jsou ve vazbě (nachází se ve vazební tabulce).

Nadpisy sloupců a řádků lze pro přehlednost realizovat jinými atributy z tabulek, které jsou ve vazbě. Například názvem zboží, jménem zákazníka atd.

Testovací data pro příklad se skládají ze 3 typů entit. Dva typy entit jsou ve vztahu M:N pomocí vazební tabulky. Jsou to firmy a zaměstnanci, kteří jsou spojeni výdejkami. Firem je v databázi 5, zaměstnanců je také 5. Ve vazební tabulce se ve vazbě vyskytují jen 2 firmy a 3 zaměstnanci. Záznamů ve vazební tabulce je také 5.

Příklad 21: Výpis hodnot primárních klíčů tabulek, které jsou spolu ve vazbě

SQL Příkaz 1:

```
SELECT id
    FROM firmy;
```

Popis SQL příkazu 1:

Příkaz vypíše všechna ID všech firem v databázové tabulce. Tato data budou použita jako nadpisy sloupců.

Výsledek SQL příkazu 1:

id
1
2
3
4
5

Tabulka 24: Výpis hodnot primárních klíčů tabulek, které jsou spolu ve vazbě příkaz 1

SQL Příkaz 2:

```
SELECT id
    FROM zamestnanec;
```

Popis SQL příkazu 2:

Příkaz vypíše všechna ID všech zaměstnanců firmy v databázové tabulce. Tato data budou použita jako nadpisy řádků.

Výsledek SQL příkazu 2:

id
1
2
3
4
5

Tabulka 25: Výpis hodnot primárních klíčů tabulek, které jsou spolu ve vazbě příkaz 2

SQL Příkaz 3:

```
SELECT id_firma, id_zamestnanec
    FROM vydejky;
```

Popis SQL příkazu 3:

Příkaz vypíše hodnoty primárních klíčů tabulek ve vazbě z vazební tabulky.

Výsledek SQL příkazu 3:

Id_firma	Id_zamestnanec
1	1
1	1
1	2
3	3
3	3

Tabulka 26: Výpis hodnot primárních klíčů tabulek, které jsou spolu ve vazbě příkaz 3

Zbytek bude řešen již programově. Bude se postupně procházet firmami a zaměstnanci, pokud jsou konkrétní hodnoty ID rovny hodnotám z vazební tabulky, vypíše se ANO, pokud daná vazba nebyla nalezena, vypíše se NE.

Příklad výsledku SQL příkazu s názvy řádků a sloupců s ID:

id_firma id_zamestnanec	1	2	3	4	5
1	ANO	NE	NE	NE	NE
2	ANO	NE	NE	NE	NE
3	NE	NE	ANO	NE	NE
4	NE	NE	NE	NE	NE
5	NE	NE	NE	NE	NE

Tabulka 27: Výpis hodnot primárních klíčů tabulek, které jsou spolu ve vazbě s ID

Příklad 22: Výpis hodnot jiných atributů než primárních klíčů z tabulek ve vazbě

V tomto případě by se změnily jen první dva příkazy select. Přidaly by se do nich atributy, které budou reprezentovat názvy sloupců. V tomto případě nazev u firmy a jmeno u zamestnance.

Příklad výsledku SQL příkazu s názvy řádků a sloupců s jiných hodnot než z primárních klíčů a s hodnotami ANO/Ne vyjádřenými pomocí fajfky a křížku:

firma zamestnanec	MORAVIA SPORT GROUP spol. s r.o.	PROMET LOGISTICS a.s.	Nakladatelství ANAGRAM s.r.o.	LEARKA a.s.	BONATRANS a.s.
Adam	✔	✘	✘	✘	✘
Pavel	✔	✘	✘	✘	✘
Rostislav	✘	✘	✔	✘	✘
Vanda	✘	✘	✘	✘	✘
Dita	✘	✘	✘	✘	✘

Tabulka 28: Výpis hodnot jiných atributů než primárních klíčů z tabulek ve vazbě

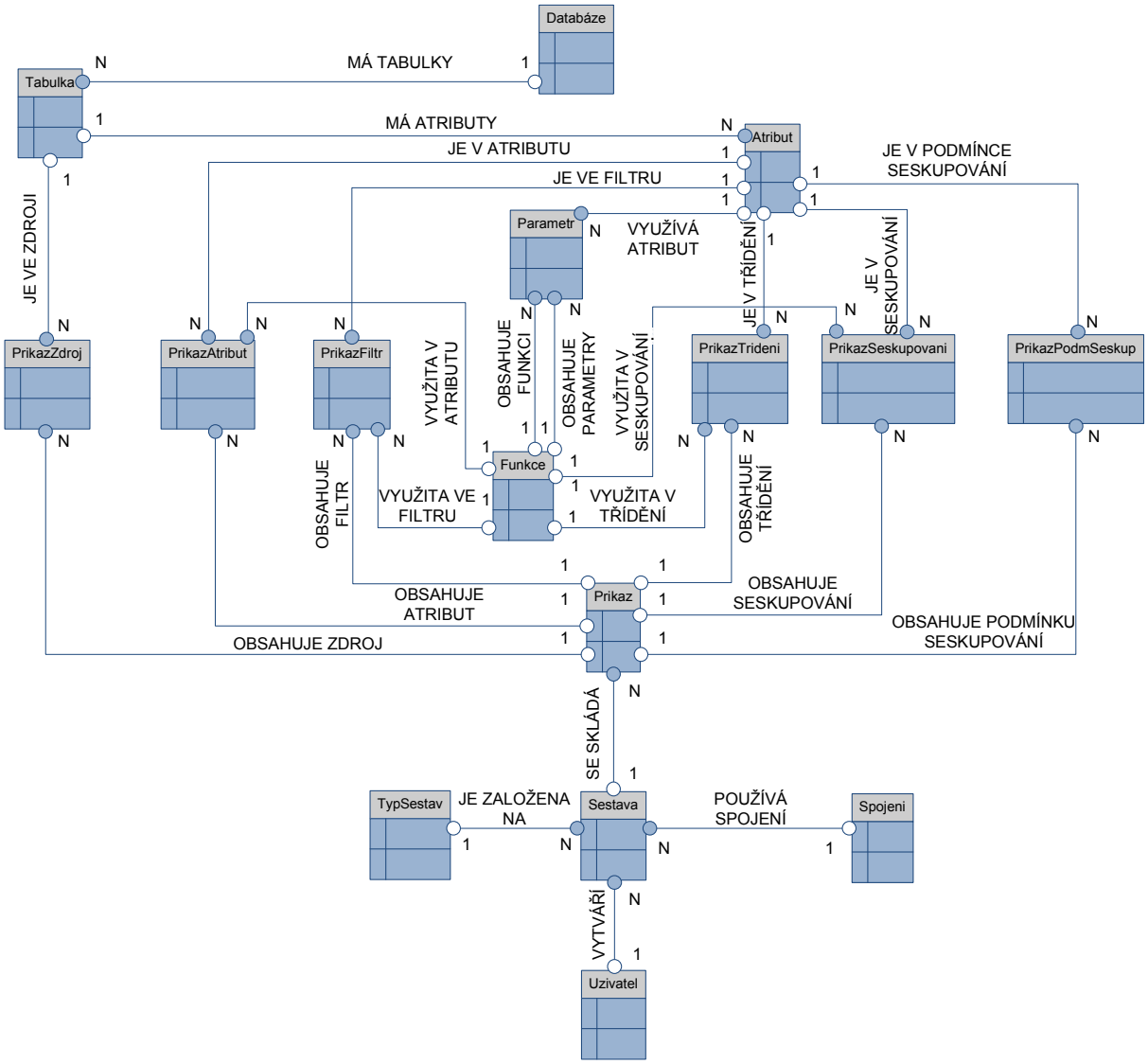
4.2 Datová analýza

V této části je navrhnuta struktura databáze, všechny evidované typy entit, jejich vzájemné vazby a jejich atributy. Také všechny datové typy všech entit.

Vzhledem k rozsáhlosti datové analýzy je zde uvedena jen její část. Celá datová analýza je uvedena v příloze č. 2 – Datová analýza.

4.2.1 E-R diagram

Níže je uveden E-R diagram s popisem typů entit a vztahů mezi nimi.



Obrázek 9: E-R diagram

4.2.2 Lineární zápis typů entit

Primární klíč, cizí klíč
Sestava (**id**, nazev, nadpis, strankovani, idSpojeni, uzivatel, idTypSestav, datum, smer, php)

TypSestav (**id**, nazev, sablona, poradi)

Spojeni (**id**, uzivatel, ip, port, instance, jmeno, heslo, typ)

Databaze (**id**, nazev)

Tabulka (**id**, idDatabaze, nazev)

Atribut (**id**, idTabulka, nazev)

Funkce (**id**, nazev)

Parametr (**id**, idFunkce, hodnota, hodnotaIdFunkce, hodnotaIdAtribut, poradi,hodnotaDatTyp)

PrikazAtribut (**id**, idAtribut, idPrikaz, idFunkce, hlavicka, poradi)

PrikazFiltr (**id**, idAtribut1, idAtribut2, idPrikaz, porovnavani, hodnota1, hodnota2, idFunkce1 , idFunkce2, logOperator, poradi, hodnota1DatTyp, hodnota2DatTyp)

PrikazSeskupovani (**id**, idAtribut, idPrikaz, poradi)

PrikazPodmSeskup (**id**, idAtribut1, idAtribut2, idPrikaz, porovnavani, hodnota1, hodnota2, idFunkce1 , idFunkce2, logOperator, poradi, hodnota1DatTyp, hodnota2DatTyp)

PrikazTrideni (**id**, idAtribut, idPrikaz, smer, idFunkce, poradi)

PrikazZdroj (**id**, idTabulka, idPrikaz, poradi)

Prikaz (**id**, idSestava, poradi, spojka)

Uzivatel (**id**, jmeno, email, heslo, blokovany, admin, otazkaNaHeslo)

4.2.3 Lineární zápis typů vztahů

MÁ TABULKY (Databaze, Tabulka) 1:N

JE VE ZDROJI (Tabulka, PrikazZdroj) 1:N

MÁ ATRIBUTY (Tabulka, Atribut) 1:N

OBSAHUJE PARAMETRY (Funkce, Parametr) 1:N

OBSAHUJE FUNKCI (Parametr, Funkce) N:1

VYUŽÍVÁ ATRIBUT (Parametr, Atribut) N:1

VYUŽITA V ATRIBUTU (Funkce, PrikazAtribut) 1:N

VYUŽITA VE FILTRU (Funkce, PrikazFiltr) 1:N

VYUŽITA V SESKUPOVÁNÍ (Funkce, PrikazPodmSeskup) 1:N

VYUŽITA V TŘÍDĚNÍ (Funkce, PrikazTrideni) 1:N

JE V ATRIBUTU (Atribut, PrikazAtribut) 1:N

JE VE FILTRU (Atribut, PrikazFiltr) 1:N

JE V TŘÍDĚNÍ (Atribut, PrikazTrideni) 1:N

JE V SESKUPOVÁNÍ (Atribut, PrikazSeskupovani) 1:N

JE V PODMÍNCE SESKUPOVÁNÍ (Atribut, PrikazPodmSeskup) 1:N

OBSAHUJE ZDROJ (Prikaz, PrikazZdroj) 1:N

OBSAHUJE ATRIBUT (Prikaz, PrikazAtribut) 1:N

OBSAHUJE FILTR (Prikaz, PrikazFiltr) 1:N

OBSAHUJE TŘÍDĚNÍ (Prikaz, PrikazTrideni) 1:N

OBSAHUJE SESKUPOVÁNÍ (Prikaz, PrikazSeskupovani) 1:N

OBSAHUJE PODMÍNKU SESKUPOVÁNÍ (Prikaz, PrikazPodmSeskup) 1:N

SE SKLÁDÁ (Sestava, Prikaz) 1:N

JE ZALOŽENA NA (Sestava, TypSestav) N:1

POUŽÍVÁ SPOJENÍ (Sestava, Spojeni) N:1

VYTVÁŘÍ (Uzivatel, Sestava) 1:N

4.2.4 Datový slovník

4.2.4.1 Typ entity PrikazZdroj

Typ entity PrikazZdroj eviduje zdroj SQL příkazu, který je uvedený za klauzulí FROM. Může jím být tabulka databáze, dočasná tabulka, pohled. Zdrojů může být více, proto se eviduje jejich pořadí v atributu poradi.

Atribut	Datový typ	Velikost	Klíč	Null	index	IO a popis
id	integer	4	ano	ne	ano	Jednoznačné číslo zdroje v příkazu
idTabulka	integer	4	ne	ne	ano	Jednoznačné číslo zdrojové tabulky, cizí klíč z typu entity Tabulka
idPrikaz	integer	4	ne	ne	ano	Jednoznačné číslo příkazu, cizí klíč z typu entity Prikaz
poradi	smallint	2	ne	ne	ne	Pořadí zdrojové tabulky v příkazu

Tabulka 29: Typ entity PrikazZdroj

4.2.4.2 Typ entity PrikazTrideni

PrikazTrideni obsahuje třídící podmínky pro SQL příkaz za klauzulí ORDER BY. U každé podmínky se eviduje, podle kterého atributu se má třídit, v jakém směru a také jaké je pořadí třídících podmínek.

Atribut	Datový typ	Velikost	Klíč	Null	index	IO a popis
id	integer	4	ano	ne	ano	Jednoznačné číslo třídění v příkazu
idAtribut	integer	4	ne	ano	ano	Jednoznačné číslo atributu v třídění, cizí klíč z typu entity Atribut
idPrikaz	integer	4	ne	ne	ano	Jednoznačné číslo příkazu, cizí klíč z typu entity Prikaz
smer	varchar	50	ne	ano	ne	Směr příkazu třídění, IO – ASC/DESC
idFunkce	integer	4	ne	ano	ano	Jednoznačné číslo funkce, cizí klíč z tabulky Funkce
poradi	smallint	2	ne	ne	ne	Pořadové číslo třídicího příkazu

Tabulka 30: Typ entity PrikazTrideni

4.2.4.3 Typ entity PrikazPodmSeskup

Tento typ entity eviduje podmínky pro seskupené řádky. Tyto podmínky se píší v příkazu SELECT za klauzulí HAVING. U každé podmínky se eviduje, jaký atribut se má porovnávat s jakou hodnotou a jakým operátorem. Podmínek v jednom příkazu může být více, proto je nutné evidovat jejich pořadí a také logickou spojku pro jejich spojování.

Atribut	Datový typ	Velikost	Klíč	Null	index	IO a popis
id	integer	4	ano	ne	ano	Jednoznačné číslo podmínky seskupování v příkazu
idAtribut1	integer	4	ne	ano	ano	Jednoznačné číslo atributu, cizí klíč z typu entity Atribut
idAtribut2	integer	4	ne	ano	ano	Jednoznačné číslo atributu, cizí klíč z typu entity Atribut
idPrikaz	integer	4	ne	ne	ano	Jednoznačné číslo příkazu, cizí klíč z typu entity Prikaz
porovnavani	varchar	50	ne	ne	ne	Porovnávací operátor pro definování podmínky, IO – (=, <>, >, <, >=, >=, BETWEEN, NOT IN, IS NULL)
hodnota1	varchar	256	ne	ano	ne	Porovnávací hodnota pro podmínku
hodnota2	varchar	256	ne	ano	ne	Porovnávací hodnota pro podmínku
idFunkce1	integer	4	ne	ano	ano	Jednoznačné číslo funkce, cizí klíč z tabulky Funkce
idFunkce2	integer	4	ne	ano	ano	Jednoznačné číslo funkce, cizí klíč z tabulky Funkce
logOperator	varchar	50	ne	ano	ne	Logické operátory pro podmínku, IO – (AND, OR)
poradi	smallint	2	ne	ne	ne	Pořadí příkazu podmínky seskupení
hodnota1DatTyp	varchar	50	ne	ano	ne	Datový typ první hodnoty
hodnota2DatTyp	varchar	50	ne	ano	ne	Datový typ druhé hodnoty

Tabulka 31: Typ entity PrikazPodmSeskup

4.3 Funkční analýza

Funkční analýza slouží jako rozbor všech funkcí analyzovaného systému. Funkce byly definovány zadavatelem nebo byly doplněny během analýzy. Všechny tyto funkce jsou umístěny v zadání.

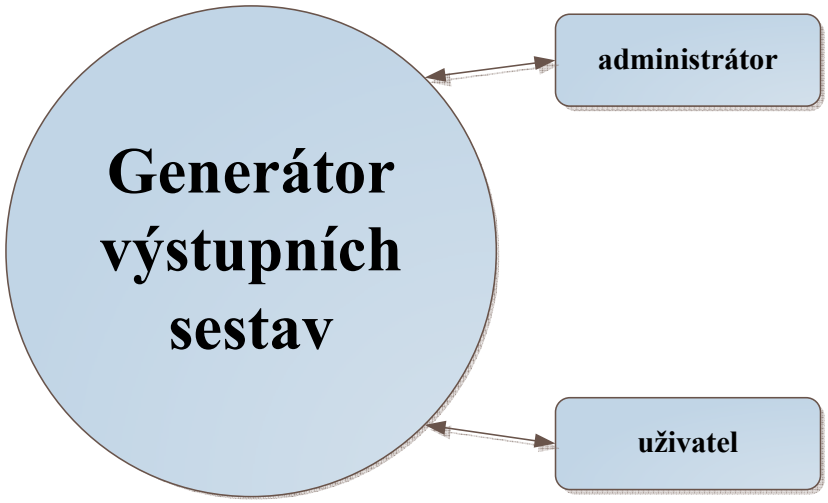
Pro popis funkcí systému, jejich propojení a jejich propojení s databází byly použity DFD diagramy a minispecifikace.

DFD diagramy, nebo li diagramy datových toků, slouží ke grafickému znázornění funkcí a toků dat mezi těmito funkcemi. Tyto diagramy jsou rozděleny do několika úrovní od abstraktního popisu celého systému až po podrobný diagram nejnižší úrovně, ve kterém jsou již popsány jednotlivé funkce.

Minispecifikace jsou slovní popis toho, co daná funkce provádí za operace a v jakém pořadí je provádí.

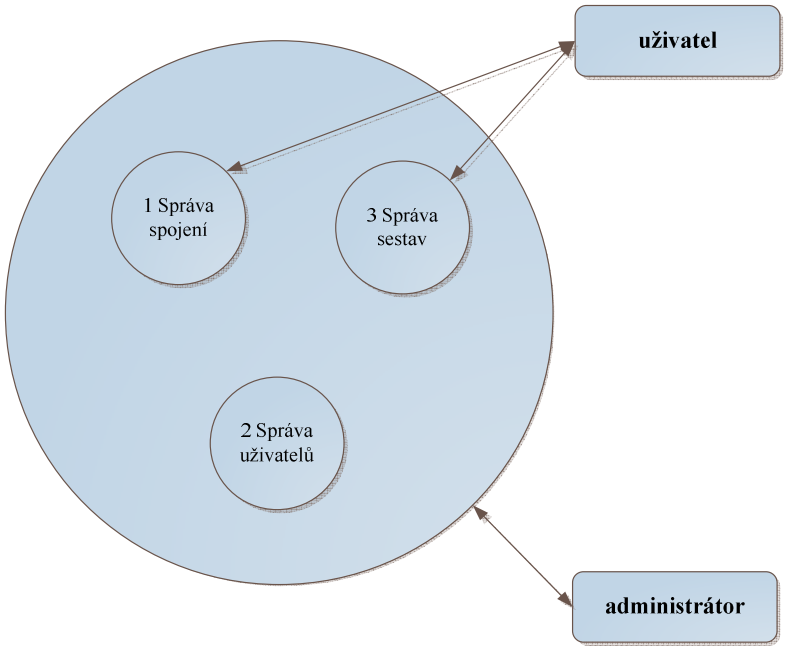
Níže je uvedena pouze část funkční analýzy. Zbytek se nachází v příloze č. 6.

4.3.1 Kontextový diagram



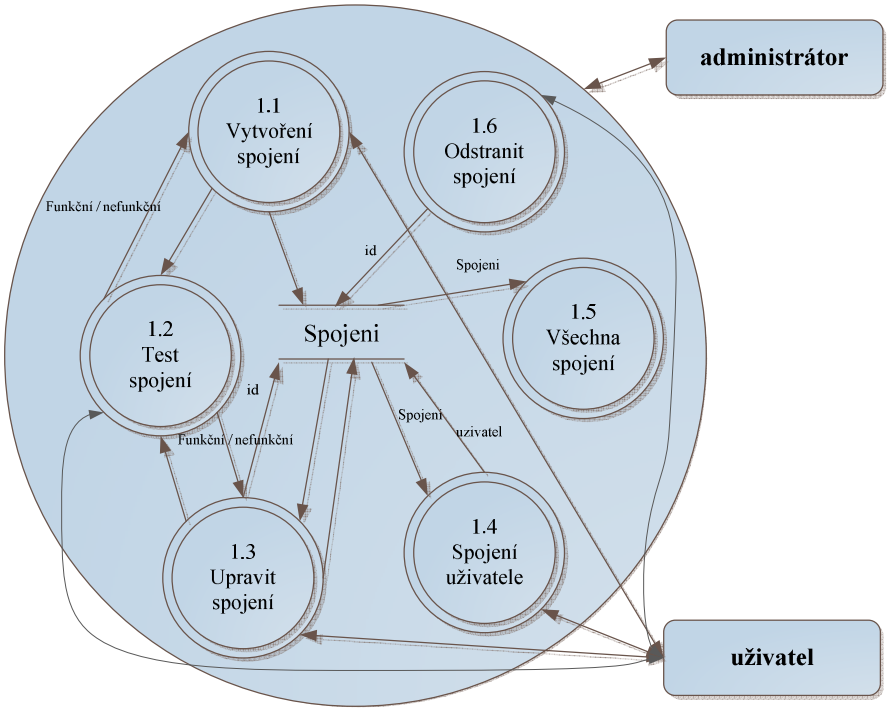
Obrázek 10: Kontextový diagram

4.3.2 DFD diagram 0. úrovně



Obrázek 11: DFD diagram 0. úrovně

4.3.3 DFD diagram 1. Úrovně: 1 Správa spojení



Obrázek 12: DFD diagram 1. Úrovně: 1 Správa spojení

4.3.3.1 Minispecifikace 1.1 Vytvoření spojení

Obrázek 13: Formulář Vytvoření spojení

Algoritmus

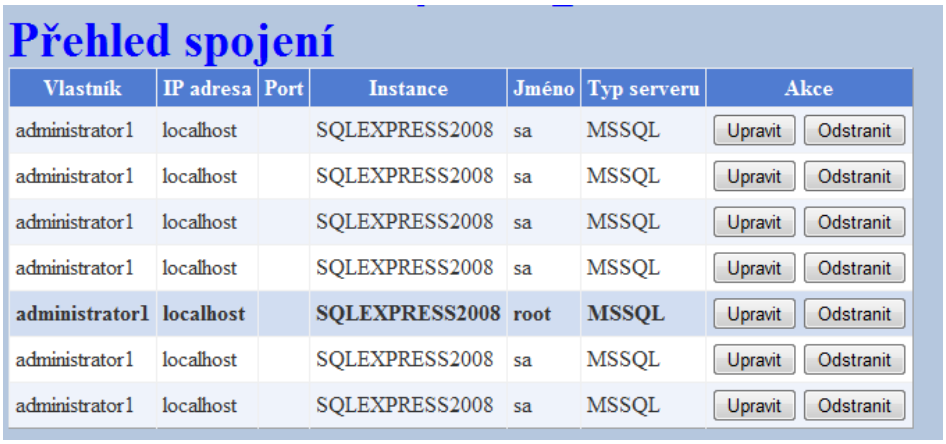
1. Zobraz formulář Vytvoření spojení
2. Uživatel vyplní potřebné údaje do paměťových proměnných
3. Zkontroluj vyplněné údaje, a pokud jsou zadány špatně, vypiš chybové hlášení a vrať se na krok 2, jinak pokračuj dále
4. Spusť funkci 1.2 Test spojení a jako parametry jí předej hodnoty zadaného spojení
5. Zjisti návratovou hodnotu funkce Test spojení, a pokud testování skončilo chybou, vypiš chybové hlášení včetně výpisu chyby spojení z této funkce a vrať se na krok 2, jinak pokračuj dále
6. Do typu entity Spojeni vlož nový záznam s příslušnými hodnotami z paměťových proměnných
7. Zobraz uživateli hlášení o úspěšném uložení spojení

4.3.3.2 Minispecifikace 1.2 Test spojení

Algoritmus

1. Pokud funkci Test spojení vyvolal sám uživatel z formuláře Vytvoření spojení, nebo Editace spojení proveď krok a, jinak, pokud se Test spojení vyvolal při ukládání, nebo editaci spojení, skoč na krok 2
    a. Zkontroluj vyplněné údaje, a pokud jsou zadány špatně, vypiš chybové hlášení a vrať se do formuláře pro Vytvoření spojení, nebo pro Editaci spojení
2. Ulož zadané parametry spojení do paměťových proměnných
3. Pokus se navázat spojení ze zadaných parametrů v paměťových proměnných
4. Pokud se spojení nepodařilo navázat, ulož chybové hlášení do paměťové proměnné a ukonči funkci s chybou, jinak pokračuj dále.
5. Pokud se spojení podařilo navázat, ukonči funkci normálně

4.3.3.3 Minispecifikace 1.3 Upravit spojení



Obrázek 14: Formulář Přehled spojení s vybraným spojením

Obrázek 15: Formulář Úprava spojení

Algoritmus

1. Pokud je přihlášen uživatel v roli administrátor, spusť funkci 1.5 Všechna spojení, jinak pokud je uživatel přihlášen v roli uživatel, spusť funkci 1.4 Spojení uživatele
2. Zobraz formulář Přehled spojení.
3. Uživatel vybere spojení, které chce upravit
4. Z databázové tabulky Spojení načti příslušný záznam a jeho hodnoty zapiš do paměťových proměnných
5. Toto spojení poté zobraz uživateli ve formuláři Úprava spojení
6. Uživatel změní hodnoty spojení
7. Zkontroluj vyplněné údaje, a pokud jsou zadány špatně, vypiš chybové hlášení a vrať se na krok 5, jinak pokračuj dále
8. Spusť funkci 1.2 Test spojení a jako parametry jí předej hodnoty zadaného spojení
9. Zjisti návratovou hodnotu funkce Test spojení, a pokud testování skončilo chybou, vypiš chybové hlášení včetně výpisu chyby spojení z této funkce a vrať se na krok 6, jinak pokračuj
10. V typu entity Spojení aktualizuj záznam s příslušným id spojení hodnotami z paměťových proměnných
11. Zobraz uživateli hlášení o úspěšném uložení spojení

4.3.3.4 Minispecifikace 1.4 Spojení uživatele

Algoritmus

1. Zjisti jméno přihlášeného uživatele
2. Podle tohoto jména najdi v databázové tabulce Spojeni všechna spojení tohoto uživatele a vypiš mu je ve formuláři Přehled spojení
3. Pokud se v databázi nenachází žádné adekvátní spojení, zobraz o tom hlášení uživateli

4.3.3.5 Minispecifikace 1.5 Všechna spojení

Algoritmus

1. Najdi v databázové tabulce Spojení všechna spojení a vypiš je ve formuláři Přehled spojení
2. Pokud se v databázi nenachází žádné adekvátní spojení, zobraz o tom hlášení uživateli

4.3.3.6 Minispecifikace 1.6 Odstranit spojení

Algoritmus

1. Pokud je přihlášen uživatel v roli administrátor, spusť funkci 1.5 Všechna spojení, jinak pokud je uživatel přihlášen v roli uživatel, spusť funkci 1.4 Spojení uživatele
2. Zobraz formulář Přehled spojení
3. Uživatel vybere spojení, které chce odstranit
4. Zobraz uživateli potvrzovací dialog, zda chce spojení skutečně smazat
5. Pokud uživatel spojení smazat nechce, skoč na krok 2, pokud jej smazat chce, pokračuj dále
6. Smaž zvolené spojení z databázové tabulky Spojení a zobraz o tom uživateli hlášení

4.4 Indexová analýza

Další důležitou částí analýzy je indexová analýza. Zabývá se vytvořením indexů pro různé atributy v databázi. Indexy urychlují vykonávání SQL příkazů hlavně pro velký počet záznamů. Indexy jsem vytvářel pro primární klíče, cizí klíče a také pro atributy, podle kterých se často vyhledává a dělají se z nich setříděné výpisy. Indexy se dělí na více typů: udržované, částečné, dočasné. Já budu používat pouze typ indexu udržovaný, protože je předpoklad, že se všechny indexované atributy budou používat často.

4.4.1 Tabulka indexová analýza

Atribut	Datový typ	Velikost	Klíč	Null	index	Důvod indexu
Typ entity Uzivatel						
id	integer	4	ano	ne	udržovaný	Primární klíč
jmeno	varchar	256	ne	ne	udržovaný	Časté výpisy, kvůli přihlašování, uloženo v jiných typech entit, třídění
admin	binary	1	ne	ne	udržovaný	Časté výpisy, přihlašování, vytváření uživatelů
Typ entity Prikaz						
id	integer	4	ano	ne	udržovaný	Primární klíč
idSestava	integer	4	ne	ne	udržovaný	Cizí klíč z typu entity Sestava
Typ entity PrikazZdroj						
id	integer	4	ano	ne	udržovaný	Primární klíč
idTabulka	integer	4	ne	ne	udržovaný	Cizí klíč z typu entity Tabulka
idPrikaz	integer	4	ne	ne	udržovaný	Cizí klíč z typu entity Prikaz
Typ entity PrikazTrideni						
id	integer	4	ano	ne	udržovaný	Primární klíč
idAtribut	integer	4	ne	ne	udržovaný	Cizí klíč z typu entity Atribut
idPrikaz	integer	4	ne	ne	udržovaný	Cizí klíč z typu entity Prikaz
idFunkce	integer	4	ne	ano	udržovaný	Cizí klíč z typu entity Funkce
Typ entity PrikazPodmSeskup						
id	integer	4	ano	ne	udržovaný	Primární klíč
idAtribut1	integer	4	ne	ano	udržovaný	Cizí klíč z typu entity Atribut
idAtribut2	integer	4	ne	ano	udržovaný	Cizí klíč z typu entity Atribut
idPrikaz	integer	4	ne	ne	udržovaný	Cizí klíč z typu entity Prikaz
idFunkce1	integer	4	ne	ano	udržovaný	Cizí klíč z typu entity Funkce
idFunkce2	integer	4	ne	ano	udržovaný	Cizí klíč z typu entity Funkce
Typ entity PrikazSeskupovani						
id	integer	4	ano	ne	udržovaný	Primární klíč
idAtribut	integer	4	ne	ne	udržovaný	Cizí klíč z typu entity Atribut
idPrikaz	integer	4	ne	ne	udržovaný	Cizí klíč z typu entity Prikaz
Typ entity PrikazFiltr						
id	integer	4	ano	ne	udržovaný	Primární klíč
idAtribut1	integer	4	ne	ano	udržovaný	Cizí klíč z typu entity Atribut
idAtribut2	integer	4	ne	ano	udržovaný	Cizí klíč z typu entity Atribut
idPrikaz	integer	4	ne	ne	udržovaný	Cizí klíč z typu entity Prikaz
idFunkce1	integer	4	ne	ano	udržovaný	Cizí klíč z typu entity Funkce

Atribut	Datový typ	Velikost	Klíč	Null	index	Důvod indexu
idFunkce2	integer	4	ne	ano	udržovaný	Cizí klíč z typu entity Funkce
Typ entity PrikazAtribut						
id	integer	4	ano	ne	udržovaný	Primární klíč
idAtribut	integer	4	ne	ano	udržovaný	Cizí klíč z typu entity Atribut
idPrikaz	integer	4	ne	ne	udržovaný	Cizí klíč z typu entity Prikaz
idFunkce	integer	4	ne	ano	udržovaný	Cizí klíč z typu entity Funkce
Typ entity Parametr						
id	integer	4	ano	ne	udržovaný	Primární klíč
idFunkce	integer	4	ne	ne	udržovaný	Cizí klíč z typu entity Funkce
hodnotaI-dAtribut	integer	4	ne	ano	udržovaný	Cizí klíč z typu entity Atribut
Typ entity Funkce						
id	integer	4	ano	ne	udržovaný	Primární klíč
Typ entity Atribut						
id	integer	4	ano	ne	udržovaný	Primární klíč
idTabulka	integer	4	ne	ne	udržovaný	Cizí klíč z typu entity Tabulka
Typ entity Tabulka						
id	integer	4	ano	ne	udržovaný	Primární klíč
idDatabaze	integer	4	ne	ne	udržovaný	Cizí klíč z typu entity Databaze
Typ entity Databaze						
id	integer	4	ano	ne	udržovaný	Primární klíč
Typ entity Spojeni						
id	integer	4	ano	ne	udržovaný	Primární klíč
Typ entity TypSestav						
id	integer	4	ano	ne	udržovaný	Primární klíč
Typ entity Sestava						
id	integer	4	ano	ne	udržovaný	Primární klíč
idSpojeni	integer	4	ne	ne	udržovaný	Cizí klíč z typu entity Spojeni
idTypSestav	integer	4	ne	ne	udržovaný	Cizí klíč z typu entity Typ se-stav

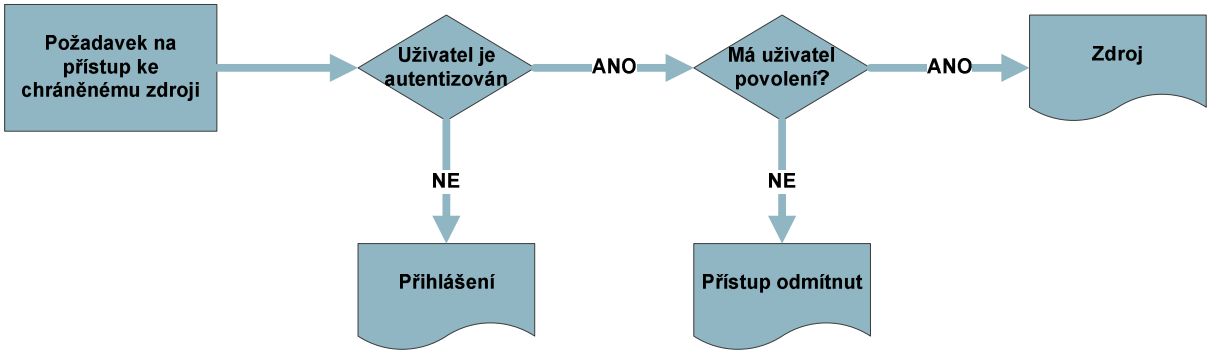
Tabulka 32: Indexová analýza

5 Návrh implementace

5.1 Návrh zabezpečení

V implementaci se budou používat pro přihlašování, evidenci uživatelů, správu rolí, obnovu hesla a další činnosti spojené se správou uživatelů integrované bezpečnostní vlastnosti ASP.NET. Tyto vlastnosti budou popsány následujícími kapitolami. Jedná se o formulářovou autentizaci, autorizaci, členství a role.

Více o možnostech zabezpečení se lze dočíst ve zdroji [20].



Obrázek 16: Žádost o přístup k webové stránce vyžadující autentizaci a autorizaci

5.1.1 Autentizace

Autentizace je proces, ve kterém se zkoumá identita uživatele a zabezpečuje se její platnost. V aplikacích ASP.NET je možné autentizaci implementovat různými metodami. Já jsem zvolil autentizaci formulářovou. Autentizace je v ASP.NET implementována pomocí speciálních http modulů. Modul je aktivní správným nastavením konfiguračního souboru. Výběr tohoto modulu se provádí v konfiguračním souboru *web.config* nacházejícím se v kořenovém adresáři aplikace pomocí prvku <authentication>.

5.1.1.1 Formulářová autentizace

Formulářová autentizace používá modul FormsAuthenticationModule. Při použití formulářové autentizace je potřeba uživatele autentizovat vůči vlastnímu úložišti. Nestačí pouze vytvořit přihlašovací stránku pro kontrolu uživatelů a hesel. Uživatele je potřeba přiřazovat do rolí a toto ASP.NET 3.5 zjednodušuje pomocí API členství, API rolí a API profilů. Při implementaci budou použity pouze API pro členství a role.

Formulářová autentizace v ASP.NET je založena na lístcích (tickets). Uživatel po přihlášení dostane lístek, obsahující jeho informace, které jsou uloženy v šifrované cookie, která je připojována k odpovědi. Když uživatel požaduje stránku, která je pro anonymního uživatele nepřístupná, runtime ASP.NET ověřuje, zda je k dispozici lístek. Pokud není, ASP.NET automaticky přesměruje uživatele na přihlašovací stránku. Při úspěšném přihlášení runtime vytvoří cookie obsahující lístek a přesměruje

uživatele na původně požadovanou stránku. Runtime zjistí, že u tohoto požadavku je autorizační cookie k dispozici a stránku zobrazí.

Výhody formulářové autentizace:

- Programátor má úplnou kontrolu nad autentizačním kódem
- Programátor má úplnou kontrolu nad vzhledem přihlašovacího formuláře
- Funguje ve všech prohlížečích
- Umožňuje rozhodnout, jak se budou ukládat uživatelské informace

Nevýhody formulářové autentizace:

- Programátor musí vytvořit vlastní uživatelské rozhraní pro přihlašování uživatelů
- Musí se udržovat katalog uživatelských přihlašovacích dokladů
- Musí se zachovávat opatrnost kvůli odposlouchávání komunikace po síti

Konfigurace formulářové autentizace

Výpis konfiguračního souboru web.config nacházejícího se v kořenovém adresáři aplikace:

```
<!--Nastavení zabezpečení aplikace. Je využita Formulářová autentizace-->
<authentication mode="Forms">
      <forms      name="GeneratorSestav"
                  loginUrl="Prihlaseni.aspx"
                  timeout="30"
                  slidingExpiration="true"
                  cookieless="AutoDetect"
                  protection="All"
                  requireSSL="false"
                  enableCrossAppRedirects="false"
                  defaultUrl="~/Zabezpecene/Uvod.aspx"
                  path="/">
      </forms>
</authentication>
```

Tato konfigurace popisuje, že je použita formulářová autentizace, kde má autentizační cookie název GeneratorSestav, přihlašovací stránka se jmenuje Prihlaseni.aspx a nachází se v kořenovém adresáři aplikace, dále, že cookie je platná pouze 30minut, po kterých dojde k odhlášení. SlidingExpiration je nastaveno tak, aby s každým požadavkem systém obnovoval platnost autentizační cookie, cookieless udává, že se bude pro práci s lístkem používat cookie, pokud to půjde. Protection nastaveno na All znamená, že autorizační cookie se bude šifrovat i podepisovat. RequireSLL je nastaveno tak, aby nemuselo být nastaveno zabezpečení SSL na webovém serveru. Dále je zakázáno přesměrování mezi aplikacemi, jako úvodní stránka je nastavena stránka Uvod.aspx nacházející se v adresáři Zabezpecene. Cesta ke cookies je nastavena na /.

Ukázka metody, která přihlašuje uživatele po stisku tlačítka Přihlásit se na přihlašovací obrazovce

```
// Metoda se vyvolá, pokud komponenta pro přihlášení chce autentifikovat
      uživatele
      protected void PrihlaseniDoAplikace(object sender,
            AuthenticateEventArgs e)
      {
      // Pokud se uživatel může přihlásit, tak jej přihlaš, jinak jej
            nepřihlašuj
            if (Membership.ValidateUser(Login1.UserName, Login1.Password))
                  e.Authenticated = true;
            else
                  e.Authenticated = false;

      }
```

Ukázka ověření, které je použito s různými podmínkami v každé metodě aplikace:

```
// Zde je hlavička metody

// Zda je uživatel přihlášen a zda je v roli administrátor, nebo uzivatel
      if (HttpContext.Current.User.Identity.IsAuthenticated &&
         (HttpContext.Current.User.IsInRole("administrator") ||
          HttpContext.Current.User.IsInRole("uzivatel")))

      {
            // Zde je tělo metody

      }
```

5.1.2 Autorizace

Autorizace je proces stanovování práv a omezení přidělených autentizovanému uživateli. Na základě informací vztahujících se k určité identitě je jí přístup k požadovaným zdrojům buď povolen, nebo odepřen. Autorizace je opět nakonfigurována v konfiguračním souboru *web.config* v příslušném adresáři aplikace a to pomocí prvku <authorization>.

Výpis konfiguračního souboru web.config nacházejícího se v adresáři Zabezpecene:

```
<authorization>
      <deny users="?"/>
      <allow roles="administrator, uzivatel"/>
      <deny users="*"/>
</authorization>
```

Tato konfigurace popisuje, že k tomuto adresáři je zamítnut přístup všem anonymním uživatelům (?), poté je povolen přístup rolím administrator a uzivatel a na konec je zakázán přístup všem ostatním (*).

Výpis konfiguračního souboru web.config nacházejícího se v adresáři Zabezpecene/Administrator:

```
<authorization>
      <deny users="?"/>
      <deny roles="uzivatel"/>
      <allow roles="administrator"/>
      <deny users="*"/>
</authorization>
```

Tato konfigurace popisuje nastavení, které nejprve zakáže přístup anonymním uživatelům, poté roli uzivatel, dále povolí přístup roli administrator a zakáže přístup všem ostatním.

5.1.3 Členství

API členství je pracovní rámec, který je nad infrastrukturou formulářové autentizace a používá se pro správu uživatelů a ověřování jejich totožnosti. Neimplementuje funkcionalitu pro autorizaci nebo správu rolí. Při použití API členství není potřeba implementovat vlastní přihlašovací obrazovky nebo úložiště pro doklady. Prostřednictvím API členství je možné vytvářet a odstraňovat uživatele programátorsky, nebo prostřednictvím webové konfigurační utility ASP.NET. Lze obnovovat hesla, ale také odesílat emaily sloužící k obnově hesla. Lze vyhledávat uživatele v podkladovém úložišti dat pro zjištění podrobností o konkrétním uživateli nebo získání seznamu uživatelů. API obsahuje mnoho ovládacích prvků pro přihlašování do aplikace, zobrazení stavu přihlášení a pro práci s uživateli. Také obsahuje vrstvu, která zabezpečuje nezávislost aplikace na podkladovém úložišti dat. To znamená, že podkladové úložiště lze nahradit jiným, aniž by se musela aplikace jakkoliv modifikovat.

5.1.3.1 Práce s API členství

Pro práci s API členství je potřebné provést jeho konfiguraci v souboru web.config a definovat připojovací řetězec k databázi poskytovatele členství, dále je třeba připravit úložiště dat pro poskytovatele členství. Nakonec už jen stačí vytvořit uživatele a umístit do aplikace ovládací prvky pro práci s API členství.

Vytvoření úložiště dat

Jako úložiště dat jsem zvolil Microsoft SQL server 2005 a vyšší. Vytvoření tohoto úložiště jsem provedl přes příkazovou řádku Windows příkazem *aspnet_regsql*. Provedení tohoto příkazu s příslušnými parametry provede vytvoření databáze, tabulek, pohledů, uložených procedur a dalších potřebných součástí pro API členství. Po provedení tohoto příkazu je úložiště členství připraveno.

Příkaz, kterým jsem vytvořil úložiště dat:

```
aspnet_regsql –S (local)\SQLEXPRESS –E –A mr –d GeneratorSestav
```

Tento příkaz vytvořil úložiště dat na lokálním SQL severu 2005 express pro role a členství v databázi GeneratorSestav. Strukturu vytvořených tabulek lze nalézt v E-R diagramu pro návrh implementace v přílohách, jako přílohu č.4, a také v příloze č.3 návrh implementace, kde je popsán kompletní datový slovník, E-R diagram i indexová analýza výsledných typů entit.

Konfigurace poskytovatele členství

Výpis konfiguračního souboru web.config nacházejícího se v kořenovém adresáři aplikace:

```
<!--Definice členství. Ověřování uživatelů z databáze-->
<membership defaultProvider="GeneratorSestavClenstviPoskytovatel">
      <providers>
            <add  name="GeneratorSestavClenstviPoskytovatel"
                  connectionStringName="GeneratorSestavPripojovaciRetezec"
                  applicationName="GeneratorSestav"
                  enablePasswordRetrieval="false"
                  enablePasswordReset="true"
                  requiresQuestionAndAnswer="true"
                  requiresUniqueEmail="true"
                  passwordFormat="Hashed"
                  maxInvalidPasswordAttempts="5"
                  minRequiredNonalphanumericCharacters="0"
                  minRequiredPasswordLength="6"
                  type="System.Web.Security.SqlMembershipProvider"/>
      </providers>
</membership>
```

Tato konfigurace popisuje, že poskytovatel členství se bude jmenovat GeneratorSestavClenstviPoskytovatel, bude využívat připojovací řetězec s názvem GeneratorSestavPropojovaciRetezec, jméno aplikace je GeneratorSestav, že heslo nepůjde získat, ale půjde obnovit. Dále je nastaveno, že bude požadováno zadání otázky a odpovědi na ni při vytváření uživatele a také zadání jedinečného emailu. Heslo bude uchováváno hashovaně a k jeho zablokování dojde při 5 neplatných pokusech o přihlášení. Heslo nemusí obsahovat žádné speciální znaky a minimální délku musí mít 6 znaků.

5.1.4 Role

Uživatelé se často seskupují do rolí. Tímto se zpřehledňuje zabezpečení webu a usnadní se tím jeho správa. ASP.NET obsahuje API pro správu rolí, které je rozšiřitelné pomocí poskytovatelů stejně jako API členství. Toto API umožňuje správu rolí, přidělování rolí uživatelům a přístup ke všem informacím o rolích v kódu. Role se využívají v autorizačních pravidlech v souborech *web.config*, nebo je k nim možný i programátorský přístup.

Konfigurace poskytovatele rolí

Výpis konfiguračního souboru web.config nacházejícího se v kořenovém adresáři aplikace:

```
<!--Definice správce rolí pro aplikace. Načte role z databáze a pracuje
s nimi-->
<roleManager      enabled="true"
                  defaultProvider="GeneratorSestavRolePoskytovatel"
                  cacheRolesInCookie="true"
                  cookieName=".roleCookie"
                  cookieTimeout="30"
                  cookieSlidingExpiration="true"
                  cookieProtection="All">
      <providers>
            <add  name="GeneratorSestavRolePoskytovatel"
                  type="System.Web.Security.SqlRoleProvider"
                  connectionStringName="GeneratorSestavPripojovaciRetezec"
                  applicationName="GeneratorSestav"/>
      </providers>
</roleManager>
```

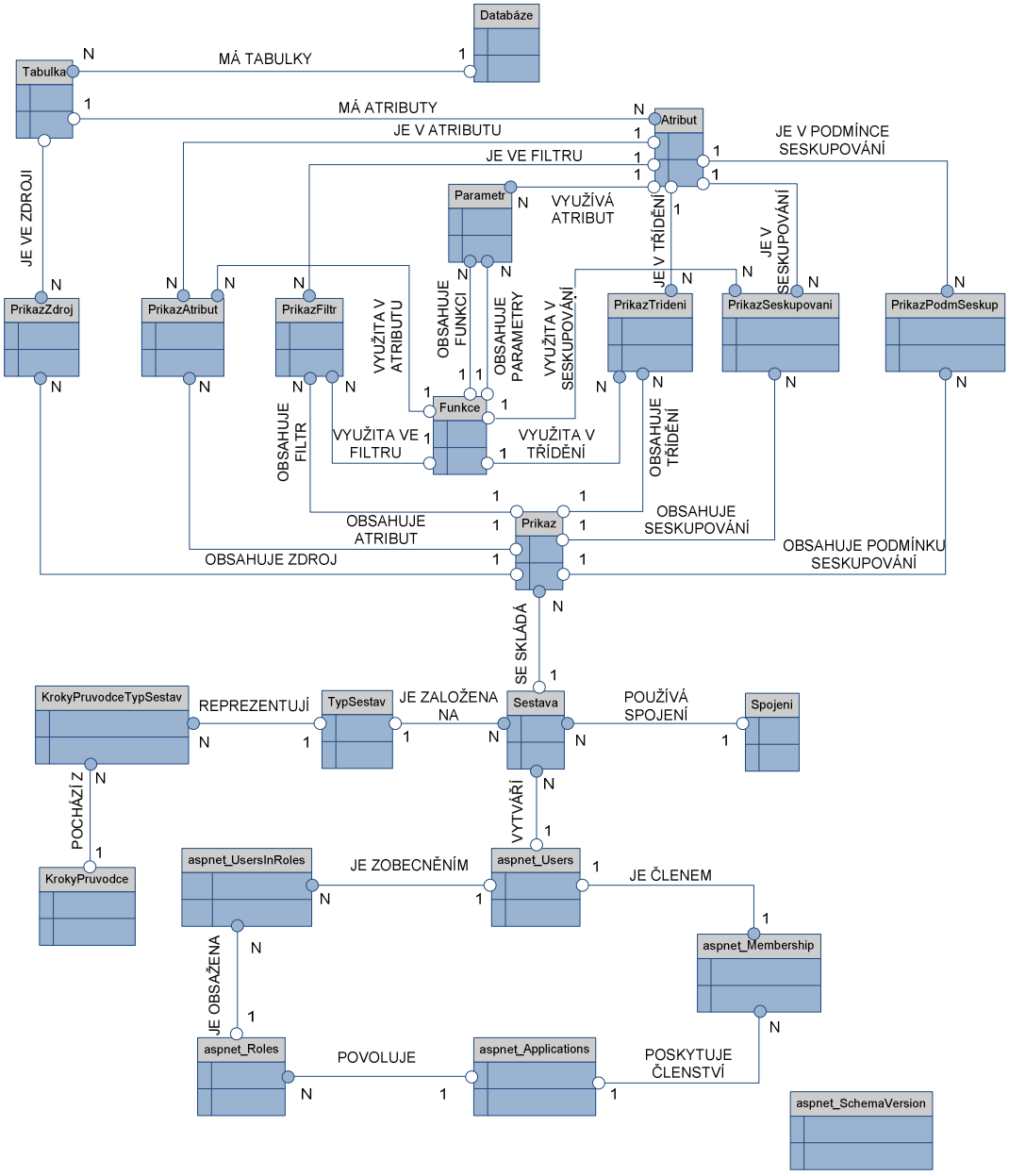
Tato konfigurace popisuje, že API rolí je aktivováno, jeho výchozím poskytovatelem je GeneratorSestavRolePoskytovatel, role jsou uchovávány v cookie, která je pojmenována .roleCookie, uchovává se po dobu 30 minut a při každém požadavku se obnovuje. Cookie je zašifrována a podepsána.

Poskytovatel je nakonfigurován s názvem GeneratorSestavRolePoskytovatel, je uložen v databázi SQL serveru, využívá příslušný spojovací řetězec a aplikace, pro kterou role poskytuje, se jmenuje GeneratorSestav.

5.2 Datová analýza

Datová analýza z kapitoly Analýza musela být upravena, protože v analýze byly u atributů v typech entit obecné datové typy a také proto, že díky využití integrovaných vlastností ASP.NET pro zabezpečení vznikly nové typy entit a ty musely být zahrnuty do návrhu databáze.Níže je zobrazen E-R diagram s novými typy entit a ukázkové typy entit. Plnohodnotný E-R diagram s popisem atributů je v příloze č. 4 a příloha č. 3 obsahuje kompletní datovou analýzu a také indexovou analýzu.

5.2.1 Databázové schéma v návrhu implementace



Obrázek 17: Databázové schéma

5.2.2 Lineární zápis typů entit

Primární klíč, cizí klíč

Sestava (**id**, nazev, nadpis, strankovani, idSpojeni, uzivatel, idTypSestav, datum, smer, php)

TypSestav (**id**, nazev, sablona, poradi)

KrokyPruvodceTypSestav (**id**, idTypSestav, idKrokyPruvodce)

KrokyPruvodce (**id**, progrId, progrTitle)

Spojeni (**id**, uzivatel, ip, port, instance, jmeno, heslo, typ)

Databaze (**id**, nazev)

Tabulka (**id**, idDatabaze, nazev)

Atribut (**id**, idTabulka, nazev)

Funkce (**id**, nazev)

Parametr (**id**, idFunkce, hodnota, hodnotaIdFunkce, hodnotaIdAtribut, poradi)

PrikazAtribut (**id**, idAtribut, idPrikaz, idFunkce, hlavicka, poradi)

PrikazFiltr (**id**, idAtribut1, idAtribut2, idPrikaz, porovnavani, hodnota1, hodnota2, idFunkce1, idFunkce2, logOperator, poradi)

PrikazSeskupovani (**id**, idAtribut, idPrikaz, smer, poradi)

PrikazPodmSeskup (**id**, idAtribut1, idAtribut2, idPrikaz, porovnavani, hodnota1, hodnota2, idFunkce1, idFunkce2, logOperator, poradi)

PrikazTrideni (**id**, idAtribut, idPrikaz, smer, idFunkce, poradi)

PrikazZdroj (**id**, idTabulka, idPrikaz, poradi)

Prikaz (**id**, idSestava, poradi, spojka)

aspnet_Membership (ApplicationId, **UserId**, Password, PasswordFormat, PasswordSalt, MobilePIN, Email, LoweredEmail, PasswordQuestion, PasswordAnswer, IsApproved, IsLockedOut, CreateDate, LastLoginDate, LastPasswordChangedDate, LastLockoutDate, FailedPasswordAttemptCount, FailedPasswordAttemptWindowStart, FailedPasswordAnswerAttemptCount, FailedPasswordAnswerWindowStart, Comment)

apsnet_Applications (ApplicationName, LoweredApplicationName, **ApplicationId**, Description)

aspnet_Users (ApplicationId, **UserId**, UserName, LoweredUserName, MobileAlias, IsAnonymous, LastActivityDate)

aspnet_UsersInRoles (**UserId**, **RoleId**)

aspnet_Roles (ApplicationId, **RoleId**, RoleName, LoweredRoleName, Description)

aspnet_SchemaVersions (**Feature**, **CompatibleSchemaVersion**, isCurrentVersion)

5.2.3 Lineární zápis typů vztahů

MÁ TABULKY (Databaze, Tabulka) 1:N

JE VE ZDROJI (Tabulka, PrikazZdroj) 1:N

MÁ ATRIBUTY (Tabulka, Atribut) 1:N

OBSAHUJE PARAMETRY (Funkce, Parametr) 1:N

OBSAHUJE FUNKCI (Parametr, Funkce) N:1

VYUŽÍVÁ ATRIBUT (Parametr, Atribut) N:1

VYUŽITA V ATRIBUTU (Funkce, PrikazAtribut) 1:N

VYUŽITA VE FILTRU (Funkce, PrikazFiltr) 1:N

VYUŽITA V SESKUPOVÁNÍ (Funkce, PrikazPodmSeskup) 1:N

VYUŽITA V TŘÍDĚNÍ (Funkce, PrikazTrideni) 1:N

JE V ATRIBUTU (Atribut, PrikazAtribut) 1:N

JE VE FILTRU (Atribut, PrikazFiltr) 1:N

JE V TŘÍDĚNÍ (Atribut, PrikazTrideni) 1:N

JE V SESKUPOVÁNÍ (Atribut, PrikazSeskupovani) 1:N

JE V PODMÍNCE SESKUPOVÁNÍ (Atribut, PrikazPodmSeskup) 1:N

OBSAHUJE ZDROJ (Prikaz, PrikazZdroj) 1:N

OBSAHUJE ATRIBUT (Prikaz, PrikazAtribut) 1:N

OBSAHUJE FILTR (Prikaz, PrikazFiltr) 1:N

OBSAHUJE TŘÍDĚNÍ (Prikaz, PrikazTrideni) 1:N

OBSAHUJE SESKUPOVÁNÍ (Prikaz, PrikazSeskupovani) 1:N

OBSAHUJE PODMÍNKU SESKUPOVÁNÍ (Prikaz, PrikazPodmSeskup) 1:N

SE SKLÁDÁ (Sestava, Prikaz) 1:N

JE ZALOŽENA NA (Sestava, TypSestav) N:1

REPREZENTUJÍ (KrokyPruvodceTypSestav, TypSestav) N:1

POCHÁZÍ S (KrokyPruvodceTypSestav, KrokyPruvodce) N:1

POUŽÍVÁ SPOJENÍ (Sestava, Spojeni) N:1

VYTVÁŘÍ (aspnet_Users, Sestava) 1:N

JE ČLENEM (aspnet_Users, aspnet_Membership) 1:1

POSKYTUJE ČLENSTVÍ (aspnet_Applications, aspnet_Membership) 1:N

POVOLUJE (aspnet_Applications, aspnet_Roles) 1:N

JE OBSAŽENA (aspnet_Roles, aspnet_UsersInRoles) 1:N

JE ZOBECNĚNÍM (aspnet_UsersInRoles, aspnet_Users) N:1

5.2.4 Datový slovník

5.2.4.1 Typ entity aspnet_Roles

Tento typ entity uchovává seznam rolí pro konkrétní aplikaci.

Atribut	Datový typ	Velikost	Klíč	Null	index	IO a popis
ApplicationId	unique-identifier	16	ne	ne	ano	Jednoznačné ID aplikace, IO – 16bitová binární hodnota zobrazená jako řetězec, cizí klíč z typu entity aspnet_Applications
RoleId	unique-identifier	16	ano	ne	ano	Jednoznačné číslo role, IO – 16bitová binární hodnota zobrazená jako řetězec
RoleName	nvarchar	256	ne	ne	ne	Název role
LoweredRoleName	nvarchar	256	ne	ne	ano	Název role (malé písmena)
Description	nvarchar	256	ne	ano	ne	Popis role

Tabulka 33: Typ entity aspnet_Roles

5.2.4.2 Typ entity aspnet_UsersInRoles

Toto je typ entity, který uchovává, kteří uživatelé patří, do kterých rolí.

Atribut	Datový typ	Velikost	Klíč	Null	index	IO a popis
UserId	unique-identifier	16	ano	ne	ano	Jednoznačné číslo uživatele, cizí klíč z typu entity aspnet_Users
RoleId	unique-identifier	16	ano	ne	ano	Jednoznačné číslo role, cizí klíč z typu entity aspnet_Roles

Tabulka 34: Typ entity aspnet_UsersInRoles

5.2.4.3 Typ entity aspnet_Users

Zde jsou uchovávány informaci o konkrétních uživatelích.

Atribut	Datový typ	Velikost	Klíč	Null	index	IO a popis
ApplicationId	unique-identifier	16	ne	ne	ano	Jednoznačné číslo aplikace, cizí klíč z typu entity aspnet_Applications
UserId	unique-identifier	16	ano	ne	ano	Jednoznačné číslo uživatele
UserName	nvarchar	256	ne	ne	ne	Uživatelské jméno
LoweredUserName	nvarchar	256	ne	ne	ano	Uživatelské jméno (malé písmena)
MobileAlias	nvarchar	16	ne	ano	ne	Přezdívka pro připojení z mobilního zařízení
IsAnonymous	bit	1	ne	ne	ne	Informace, zda se může uživatel přihlásit anonymně, IO (0/1)
LastActivityDate	datetime	8	ne	ne	ano	Datum poslední aktivity, IO – RRRR-MM-DD HH:MM:SS.MIMIMI

Tabulka 35: Typ entity aspnet_Users

5.2.4.4 Typ entity PrikazZdroj

Typ entity PrikazZdroj eviduje zdroj SQL příkazu, který je uvedený za klauzulí FROM. Může jím být tabulka databáze, dočasná tabulka, pohled. Zdrojů může být více, proto se eviduje jejich pořadí v atributu poradi.

Atribut	Datový typ	Velikost	Klíč	Null	index	IO a popis
id	unique-identifier	16	ano	ne	ano	Jednoznačné číslo zdroje v příkazu
idTabulka	unique-identifier	16	ne	ne	ano	Jednoznačné číslo zdrojové tabulky, cizí klíč z typu entity Tabulka
idPrikaz	unique-identifier	16	ne	ne	ano	Jednoznačné číslo příkazu, cizí klíč z typu entity Prikaz
poradi	smallint	2	ne	ne	ne	Pořadí zdrojové tabulky v příkazu

Tabulka 36: Typ entity PrikazZdroj

5.2.4.5 Typ entity PrikazTrideni

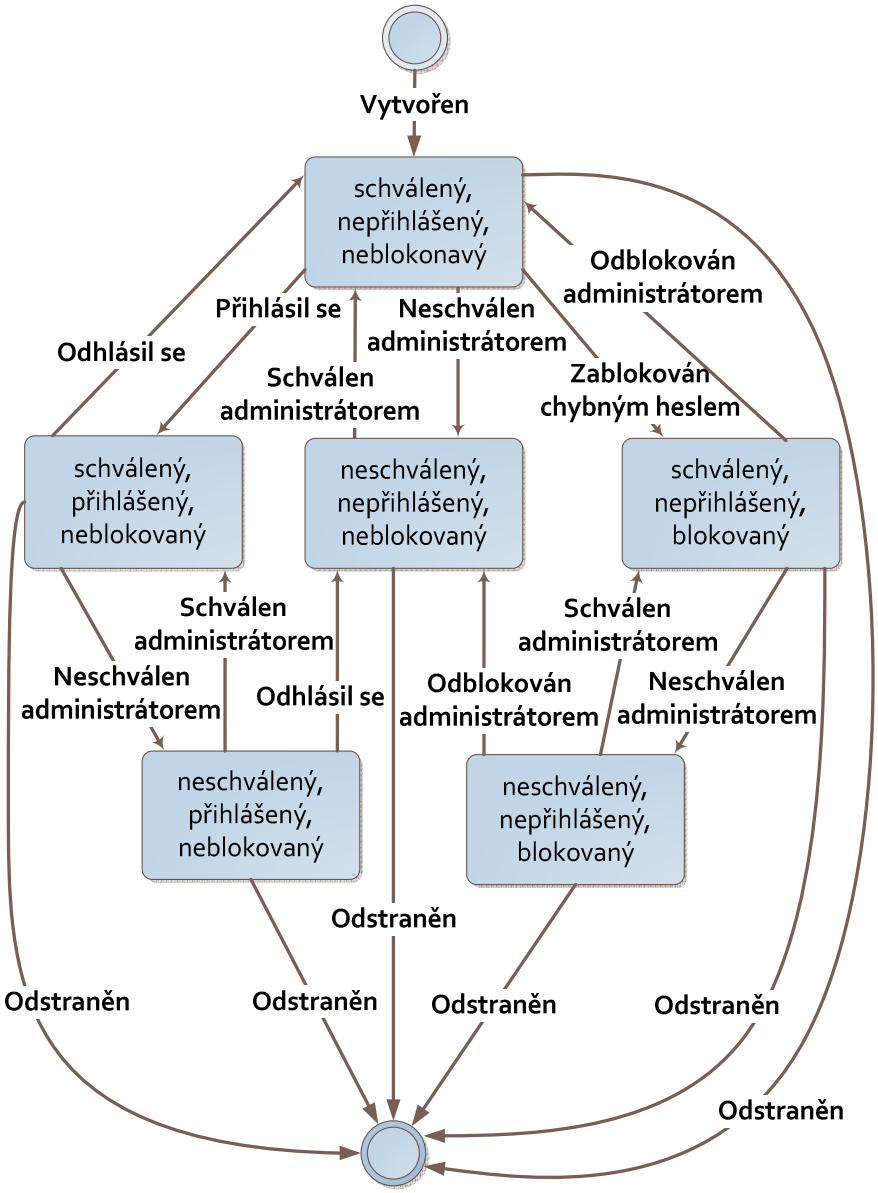
PrikazTrideni obsahuje třídící podmínky pro SQL příkaz za klauzulí ORDER BY. U každé podmínky se eviduje podle kterého atributu se má třídit, v jakém směru a také jaké je pořadí třídících podmínek.

Atribut	Datový typ	Velikost	Klíč	Null	index	IO a popis
id	unique-identi-fier	16	ano	ne	ano	Jednoznačné číslo třídění v příkazu
idAtribut	unique-identi-fier	16	ne	ano	ano	Jednoznačné číslo atributu v třídění, cizí klíč z typu entity Atribut
idPrikaz	unique-identi-fier	16	ne	ne	ano	Jednoznačné číslo příkazu, cizí klíč z typu entity Prikaz
smer	nvar-char	50	ne	ano	ne	Směr příkazu třídění, IO – ASC/DESC
idFunkce	unique-identi-fier	16	ne	ano	ano	Jednoznačné číslo funkce, cizí klíč z tabulky Funkce
poradi	smallint	2	ne	ne	ne	Pořadové číslo třídicího příkazu

Tabulka 37: Typ entity PrikazTrideni

5.3 Stavová analýza

Vzhledem k použití integrovaných vlastností ASP.NET pro zabezpečení aplikace vznikly nové typy entit, které jsou popsány v E-R diagramu výše . V těchto typech entit se objekt uživatel rozdělil do více typů entit a díky tomuto také vznikly u objektu uživatel stavy, mezi kterými přechází. V typu entity aspnet_Membership jsou tři binární atributy, které tyto stavy udávají. Uživateli bude buď povoleno přihlašování, nebo bude zakázán z důvodu chybného zadání hesla, nebo bude zakázán administrátorem pro přihlášení, nebo mohou tyto dva stavy nastat současně. Mohou nastat ještě další dva stavy, ale ty jsou popsány v diagramu. Přechody mezi stavy jsou zobrazeny na stavovém diagramu níže.



Obrázek 18: Stavový diagram uživatele

6 Implementace

Hlavním zdrojem, ze kterého jsem čerpal při implementaci, byla kniha ASP.NET 3.5 a C# 2008 [20], která je velmi podrobná a vysvětluje všechny důležité součásti těchto dvou jazyků. Dále jsem čerpal z internetových diskusí, jako je diskuse přímo na oficiálních stránkách firmy Microsoft věnující se ASP.NET [21] a z dalších stránek věnujících se řešení specifických problémů při implementaci ASP.NET webových stránek. Neméně důležitým zdrojem při implementaci byla MSDN knihovna firmy Microsoft [22]. Zde lze nalézt popis všech komponent .NET, jejich tříd, metod i argumentů. Je to velice obsáhlá knihovna, která obsahuje i ukázkové příklady.

6.1 Požadavky

Při výběru programovacích jazyků a databází jsem nebyl omezen. Nebyl kladen důraz na to, aby vývojové prostředí a databázové servery byly zdarma nebo aby byly multiplatformní. Proto jsem zvolil implementaci jako webovou stránku ASP.NET s pomocí jazyka C# ve vývojovém prostředí Visual Studio 2008. K tomuto vývojovému prostředí mám k dispozici školní licenci. Data aplikace jsou umístěna na Microsoft SQL serveru 2008 Enterprise 32bit, ke kterému mám také školní licenci. Tento server nemusí být umístěn na stejném stroji jako webová stránka. Databázové servery, ze kterých mohou uživatelé vytvářet spojení a sestavy jsou Oracle express edition, který je zdarma, dále MySQL, který je také zdarma a posledním je Microsoft SQL server. Tyto databázové servery nemusí být umístěny na stejném serveru jako aplikace. Jako operační systém na serveru jsem použil Microsoft Windows. Opět se školní licencí. V této kapitole se omezím pouze na 32bitové verze aplikací.

Tomuto výběru budou přizpůsobeny i minimální softwarové a hardwarové požadavky, které jsou popsány níže.

6.1.1 Minimální hardwarové požadavky

6.1.1.1 Serverová část

Webový server

Požadavky na webový server se liší podle toho, jaká verze webového serveru bude nainstalována a na jakém operačním systému. Níže uvedu požadavky na webový server IIS 7, který se nachází v operačním systému Windows Vista, Windows 7, nebo ve Windows Server 2008.

IIS 7 na Windows 7 (32bit)

- Procesor s frekvencí 1.0 GHz, nebo vyšší
- 1 GB paměti RAM
- 16GB místa na HDD

Databázový server

Databázovým serverem může být Microsoft SQL server od verze 2005 bez SP do verze 2008 se SP a to jak ve verzi Express, tak i ve verzi Enterprise. Níže uvedu požadavky na Microsoft SQL server 2008, na kterém probíhal vývoj aplikace. Více o minimálních požadavcích na Microsoft SQL Server se lze dočíst v *tomto zdroji* [23].

Microsoft SQL server 2008 Enterprise (32bit)

- Pentium III kompatibilní procesor na frekvenci alespoň 1.0 GHz
- 512 MB paměti RAM
- CD/DVD mechanika pro instalaci z kompaktního disku
- VGA grafická karta pro grafické nástroje serveru
- Polohovací zařízení (myš, touchpad, atd.)

6.1.1.2 Klientská část

- Připojení k internetu (pokud není klientem webový server)
- Polohovací zařízení
- Zobrazovací zařízení

6.1.2 Minimální softwarové požadavky

6.1.2.1 Serverová část

Webový server

- Operační systém Windows XP, nebo Server 2003 a vyšší
- ODBC ovladače (dodávané na přiloženém DVD v adresáři **/Aplikace/ODBC**)
- .NET Framework 3.5 SP1
- IIS server 6.0

Databázový server

Microsoft SQL server 2008 Enterprise (32bit)

- Operační systém Windows XP SP2 a vyšší, nebo Windows Server 2003 a vyšší
- .NET Framework 3.5 SP1, SQL server native Client, SQL Server Setup support files – pokud tyto komponenty nejsou nainstalovány, jejich instalace se provede na začátku instalace SQL serveru.
- Microsoft Windows Installer 4.5, nebo aktuálnější
- Podporovaný operační systém se síťovým rozhraním
- Microsoft Internet Explorer 6 SP1, nebo vyšší

6.1.2.2 Klientská část

- Internetový prohlížeč (Internet explorer, Firefox, Opera, Safari, …)

6.2 Instalace

Pro správný chod aplikace je třeba mít nainstalovaný webový server. V tomto případě IIS od firmy Microsoft. Dále je třeba na webovém serveru mít nainstalovaný .NET Framework 3.5. Další důležitou součástí je databázový server, kterým je v tomto případě Microsoft SQL Server 2005 a vyšší. Po nainstalování těchto serverů je nutné je oba nastavit a také provést nastavení aplikace.

Popis instalace obou serverů a jejich konfigurace je velmi podrobně popsána v uživatelské příručce k aplikaci nacházející se na přiloženém DVD v adresáři **/Prirucky/Uzivatelska**.

6.3 Přihlašování do aplikace

Při spuštění aplikace (zadáním příslušné URL adresy do internetového prohlížeče) se zobrazí přihlašovací stránka, která se nachází na obrázku níže. Její popis následuje pod obrázkem.

Obrázek 19: Přihlašovací stránka

Do prvního pole se zadává jméno uživatele a do druhého pole se zadává heslo pro uživatele. Heslo se z důvodu bezpečnosti nezobrazuje, jeho znaky jsou nahrazeny hvězdičkami, nebo puntíky podle použitého internetového prohlížeče. Dále obsahuje přihlašovací formulář zaškrtávací políčko Zapamatovat si přihlašovací údaje. Pokud uživatel toto pole zaškrtne, dojde k zapamatování přihlašovacích údajů v cookies internetového prohlížeče, ve kterém je aplikace spuštěna. Po zadání přihlašovacích údajů uživatel stiskne tlačítko Přihlásit. Poté dojde k ověření zadaných informací a v případě, že tyto údaje jsou správné, provede se přihlášení a přesměrování na úvodní stránku aplikace. Pokud uživatel nezadá žádné údaje a pokusí se přihlásit, zobrazí se vedle polí červená hvězdička a přihlašování se neprovede. Pokud uživatel zadá nesprávné přihlašovací údaje, zobrazí se mu chybové hlášení a přihlášení se opět neprovede. Pokud uživatel zadá 5x za sebou nesprávné heslo, dojde k zablokování jeho uživatelského účtu a již nebude schopen se do aplikace přihlásit. Odblokování uživatelského účtu je poté možné pouze po přihlášení přes jiného uživatele, který má administrátorská práva a změní hodnotu zablokován u zablokovaného uživatele.

V případě, že by uživatel zapomněl své heslo, je zde možnost jeho obnovy pomocí odkazu Obnova hesla. Tato funkce systému je popsána v uživatelské příručce, která je k dispozici na přiloženém DVD v adresáři **/Prirucky/Uzivatelska.**

Po instalaci aplikace a databáze k ní příslušné jsou vytvořeni v této databázi dva uživatelé. První uživatel má jméno **uzivatel1** a heslo **uzivatel1** a patří do role **uživatel**. To znamená, že nemá přístup ke správě uživatelů v aplikaci. Druhým uživatelem je uživatel **administrator1** s heslem **administrator1**, který patří do role **administrátor**. Pod těmito dvěma uživateli je možné se přihlásit do aplikace a poté jim změnit údaje jako heslo, email, případně tyto uživatele zablokovat nebo vymazat ze systému.

6.4 Použité programové prostředky

6.4.1 Vývoj aplikace

- Operační systém: Microsoft Windows XP Professional SP3
- Webový server: Microsoft ASP.NET Development Server
  Apache 2.0.63
- Databázový server: MySQL 5.0.51b
  Oracle 10g express edition
  Microsoft SQL Server 2008 Enterprise SP1
- ODBC ovladače MySQL ODBC 5.1 Driver
  Oracle in instantclient_11_1
  SQL Server Native Client 10.0
- Internetový prohlížeč: Microsoft Internet Explorer 6 SP3
  Mozilla Firefox 3.6
- Verze jazyků: PHP 5.2.6
  .NET Framwork 3.5 SP1
- Vývojové prostředí: Microsoft Visual Studio 2008 SP1
  PSPad 4.5.4
- Ostatní nástroje: PhpMyAdmin 2.11.7
  SQL Server Management Studio
  MySQL Administrator

6.4.2 Testování aplikace

6.4.2.1 Serverová část

- Operační systém:
  - Microsoft Windows XP Professional SP3
  - Microsoft Windows Server 2008
  - Microsoft Windows Vista Home Premium 32bit SP2
  - Microsoft Windows 7 Professional 64bit
- Webový server:
  - Microsoft ASP.NET Development Server
  - Microsoft IIS 7
  - Microsoft IIS 6
  - Apache 2.0.63
- Databázový server:
  - MySQL 5.0.51b
  - Oracle 10g express edition
  - Microsoft SQL Server 2008 Enterprise SP1
  - Microsoft SQL Server 2008 Express
  - Microsoft SQL Server 2005 Express
- Verze jazyků:
  - PHP 5.2.6
  - .NET Framwork 3.5 SP1
- ODBC ovladače
  - MySQL ODBC 5.1 Driver
  - Oracle in instantclient_11_1
  - SQL Server Native Client 10.0

6.4.2.2 Klientská část

- Operační systém:
  - Microsoft Windows XP Professional SP3
  - Microsoft Windows Vista Home Premium 32bit SP2
  - Microsoft Windows Vista Business 32bit SP1
  - Microsoft Windows 7 Professional 64bit
- Internetový prohlížeč:
  - Microsoft Internet Explorer 6 SP3
  - Microsoft Intenet Explorer 8
  - Mozilla Firefox 3.6
  - Opera 10.51
  - Apple Safari 4.0.5

6.4.3 Sestavení dokumentace

- Textový editor: Microsoft Word 2007 + doplněk SaveAsPDF
- Editor diagramů: Microsoft Visio 2007
- Grafický editor: Gimp 2.6, Microsoft malování, Microsoft výstřižky

6.5 Grafický návrh

Výsledná webová aplikace Generátor sestav využívá pro formátování stránek a prvků na nich vlastností prvků ASP.NET. Jejich vzhled je formátován pomocí motivu, vzorů stránek a také CSS stylů.

6.5.1 Kaskádové stylové předpisy (CSS styly)

Kaskádové styly umožňují aplikovat standardizované formátování na prvky webové stránky. Toto formátování přesahuje hranice platforem, protože jej lze aplikovat na HTML i XHTML a podporují jej prakticky všechny moderní internetové prohlížeče. Při práci s CSS se definuje sada pravidel, která jsou součástí předpisu. Tento předpis se později aplikuje na ovládací prvek nastavením předpisu do vlastnosti CssClass u prvku ASP.NET, do vlastnosti class u prvku HTML, nebo do vlastnosti BackgroundClass u prvku AJAX.

CSS styly jsem použil hlavně pro formátování tabulek a dialogových oken aplikace.

6.5.1.1 Příklad stylového předpisu a jeho aplikace v aplikaci

Předpis:

```
.modalBackground
{
    background-color: Gray;
    opacity: 0.7;
    filter: alpha(opacity=70);
    -moz-opacity: 0.7;
    -khtml-opacity: 0.7;
}
```

Aplikace CSS stylu v ovládacím prvku:

```
<%--Dialogové okno pro potvrzení odhlášení--%>
    <cc1:ModalPopupExtender ID="ModalPopupExtender1" runat="server"
DropShadow="True" TargetControlID="Button1" PopupControlID="popupPanel"
BackgroundCssClass="modalBackground" />
```

6.5.2 Motivy

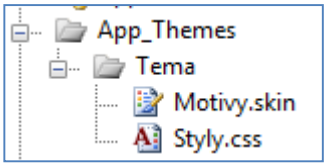
CSS styly neumožňují provádět pokročilé formátování nebo nastavování chování objektů v ASP.NET, mají jen omezenou sadu atributů stylu: Například nelze CSS styly naformátovat u prvku CheckBox-List způsob uspořádání položek (řádky, sloupce) nebo způsob zalamování textu v textových polích. Tyto mezery vyplňují motivy. Motivy nenahrazují CSS styly, ale reprezentují model vyšší úrovně.

Hlavním rozdílem je ovšem to, že motivy nejsou implementovány webovým prohlížečem, ale jsou implementovány na serveru. Motivy jsou založeny na ovládacích prvcích a ne na HTML, tímpádem umožňují definovat téměř všechny vlastnosti ovládacího prvku. Motivy se dají aplikovat prostřednictvím konfiguračního souboru *web.config* v kořenovém adresáři aplikace, protože se vztahují k celé aplikaci. Pokud se v aplikaci použije motiv, musí se pro něj vytvořit adresář v adresáři *App_Theme* aplikace. Motiv je soubor s příponou *.skin*.

6.5.2.1 Ukázka motivu v aplikaci

Adresář s motivem

Na obrázku je uveden adresář s motivem nazvaným Tema. Soubor Motivy.skin obsahuje motivy jednotlivých ovládacích prvků a soubor Styly.css obsahuje definici kaskádových stylů.



Obrázek 20: Adresář s motivem

Konfigurační soubor web.config v kořenovém adresáři aplikace

```
<!--Definice vzhledu aplikace pomocí témat-->
<pages theme="Tema">
      <controls>
            <add tagPrefix="asp" namespace="System.Web.UI"
            assembly="System.Web.Extensions, Version=3.5.0.0,
            Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
            <add tagPrefix="asp" namespace="System.Web.UI.WebControls"
            assembly="System.Web.Extensions, Version=3.5.0.0,
            Culture=neutral, PublicKeyToken=31BF3856AD364E35"/>
      </controls>
</pages>
```

Zde je nastaveno jako téma aplikace Tema.

Ukázka tématu jednoho ovládacího prvku definovaného v souboru Motivy.skin

```
<asp:Login  SkinID="Prihlaseni" runat="server" BackColor="#EFF3FB"
            BorderColor="#B5C7DE" BorderPadding="4" BorderStyle="Solid"
            BorderWidth="1px" Font-Names="Verdana" Font-Size="0.8em"
            ForeColor="#333333" Height="142px"
            style="text-align: center" Width="342px">
      <TextBoxStyle Font-Size="0.8em" />
      <LoginButtonStyle BackColor="White" BorderColor="#507CD1"
            BorderStyle="Solid" BorderWidth="1px" Font-Names="Verdana"
            Font-Size="0.8em" ForeColor="#284E98" />
      <InstructionTextStyle Font-Italic="True" ForeColor="Black" />
      <TitleTextStyle BackColor="#507CD1" Font-Bold="True"
            Font-Size="0.9em" ForeColor="White" />
</asp:Login>
```

Přiřazení tématu k ovládacímu prvku

```
<%--Komponenta pro přihlášení uživatele do aplikace, umí si pamatovat
přihlášení nebo obnovovat heslo
Nastavení jejího vzhledu, vlastností a událostí--%>
<asp:Login SkinID="Prihlaseni" ID="Login1" runat="server"
      OnAuthenticate="PrihlaseniDoAplikace"
      PasswordRecoveryText="Obnova hesla"
      PasswordRecoveryUrl="~/ObnovaHesla.aspx"
      FailureText="Pokus o přihlášení nebyl úspěšný. Opakujte akci."
      LoginButtonText="Přihlásit" PasswordLabelText="Heslo:"
      PasswordRequiredErrorMessage="Heslo je povinné."
      RememberMeText="Zapamatovat si přihlašovací údaje"
      TitleText="Přihlásit"
      UserNameLabelText="Uživatelské jméno:"
      UserNameRequiredErrorMessage="Uživatelské jméno je povinné.">
</asp:Login>
```

6.5.3 Vzory

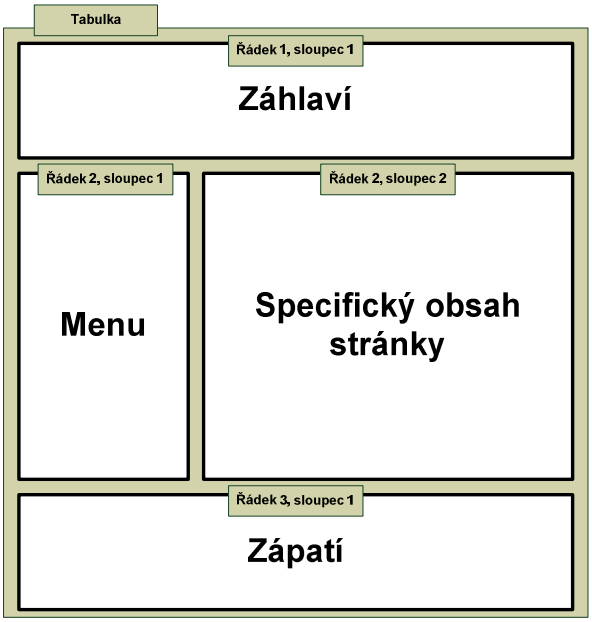
Vzory stránek byly navrženy pro potřeby standardizace layoutu (rozvržení) webových stránek. Jsou to šablony pro webovou stránku, které deklarují fixní obsah a část, kam se může vkládat vlastní obsah. Pokud se ve webové aplikaci použije jeden vzor stránek, zajistí se tím stejné rozložení prvků v celé aplikaci. Pokud se změní vzor stránek, změní se automaticky také na všech stránkách, kde se daný vzor použil. Vzory stránek poskytují nejenom systém pro opětovné využívání šablon, ale také pro jejich modifikaci a podporu jejich návrhu.

ASP.NET definuje dva typy stránek: stránky sloužící jako vzor a stránky s obsahem. Stránka sloužící jako vzor (master page) je šablona stránky. Tato stránka obsahuje, podobně jako obyčejná, HTML kód, ovládací prvky či programový kód. Navíc zde mohou být umístěni i zástupci obsahu (kontent placeholders), což jsou místa, která se mohou modifikovat. Každá obsahová stránka (content page) se odkazuje na jiný vzor stránek, přičemž z něj přebírá jak rozvržení, tak i obsah. Obsahová stránka může navíc doplňovat části, které vzor stránek nedefinuje.

6.5.3.1 Vzor stránek použitý v aplikaci

Aplikace používá dva vzory stránek. Jeden pro část webové aplikace přístupné bez přihlašování a druhý vzor pro obsah viditelný pouze pro přihlášení. V další části budu popisovat pouze vzor viditelný pro přihlášení.

Vzor stránky je založen na formátování pomocí HTML tabulky a to podle následujícího schématu.



Obrázek 21: Schéma rozložení stránky

Zjednodušený zdrojový kód stránky sloužící jako vzor

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="Hlavni2.master.cs" Inherits="generator.Zabezpecene.Hlavni2" %>
<%--Hlavička HTML dokumentu--%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
      <title>Generátor výstupních sestav</title>
</head>
<%--Tělo stránky--%>
<body class="pozadi">
   <form id="form1" runat="server">
   <asp:scriptmanager ID="Scriptmanager1"
      runat="server"></asp:scriptmanager>
   <table class="style1">
      <tr>
         <td class="style3" colspan="2">
            <%--Hlavní nadpis stránky--%>
            <asp:Label SkinID="Nadpis1" ID="Label1" runat="server"
            Text="Generátor výstupních sestav"></asp:Label>
         </td>
      </tr>
      <tr>
         <td class="style4_2">
            <%--Menu v levé částu stránky obsahující všechny funkce--%>
            <asp:Menu SkinID="Menu" ID="Menu1" runat="server"
            DataSourceID="SiteMapDataSource1"
            OnMenuItemDataBound="NacteniPolozkyMenu">
            </asp:Menu>
            <%--Zdroj dat pro menu--%>
            <asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server"
            ShowStartingNode="false" />
         </td>
         <td class="style5_2">
            <%--Obsah okna--%>
            <asp:ContentPlaceHolder ID="ContentPlaceHolder1"
            runat="server">
            </asp:ContentPlaceHolder>
         </td>
      </tr>
      <tr>
         <td class="style2" colspan="2">
            <asp:Label SkinID="Paticka" ID="Label2" runat="server"
            Text="Autor: Bc. Adam Bartoníček"></asp:Label>
         </td>
      </tr>
   </table>
   </form>
</body>
</html>
```

Na tomto kódu je znázorněno vytvoření vzorové stránky, která obsahuje hlavičku, menu, obsahovou část (ContentPlaceHolder) a patičku. Tedy přesně to rozložení, které je znázorněno na obrázku před zdrojovým kódem.

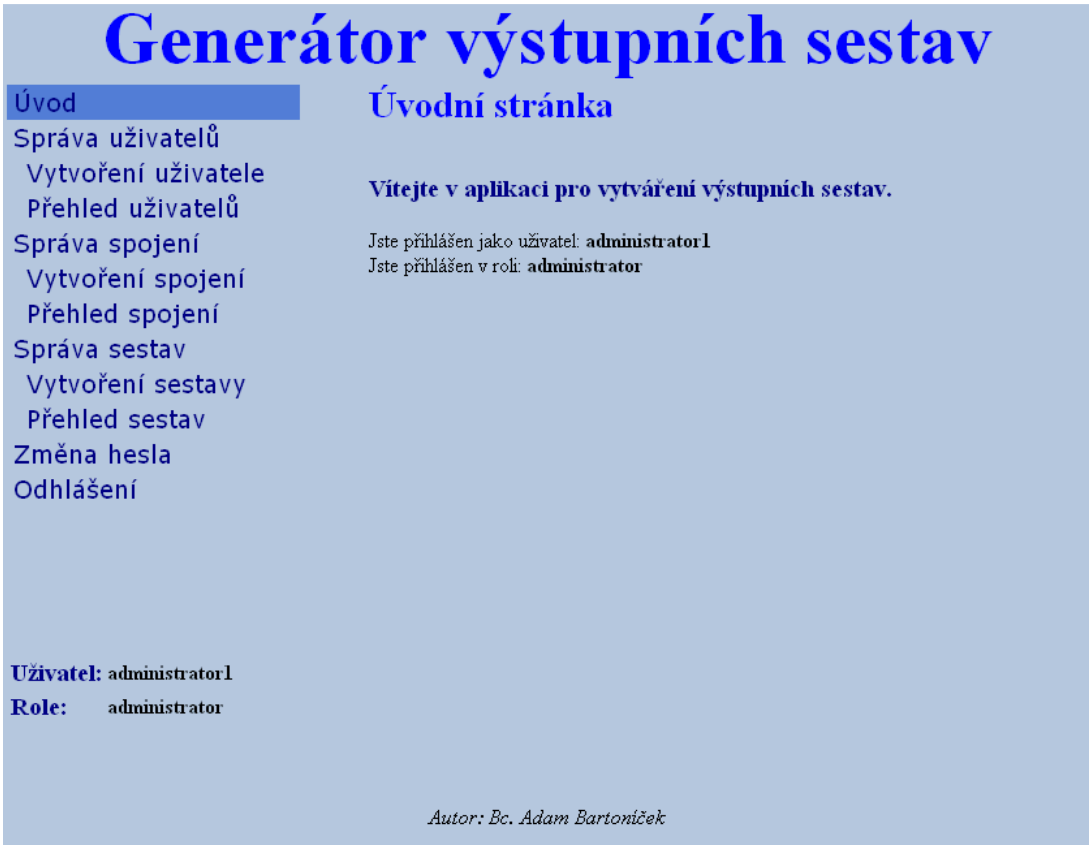
Zdrojový kód obsahové stránky

```
<%@ Page Title="Generátor sestav - Úvodní stránka" Language="C#"
MasterPageFile="~/Zabezpecene/Hlavni2.master"
AutoEventWireup="true" CodeFile="Uvod.aspx.cs"
Inherits="generator.Zabezpecene.Uvod" %>
<%--Úvodní stránka s uvítáním v aplikaci a zobrazení informací o
přihlášeném uživateli--%>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
runat="server">
      <asp:Label SkinID="Nadpis2" ID="Label2" runat="server" Text="Úvodní
            stránka"></asp:Label>
      <br />
      <br />
      <br />
      <asp:Label SkinID="Nadpis3" ID="Label4" runat="server"
            Text=""></asp:Label>
      <br />
      <br />
      <asp:Label ID="Label1" runat="server" Text=""></asp:Label>
      <br />
      <asp:Label ID="Label3" runat="server" Text=""></asp:Label>
</asp:Content>
```

Na tomto kódu je znázorněno použití vzoru stránek a obsahového prvku. Je vidět, že obsahová stránka již neobsahuje HTML hlavičku ani definici formuláře. Tato stránka je při zpracovávání požadavku na ni vložena do vzoru stránek.

Uživatelské rozhraní bylo navrhováno tak, aby bylo intuitivní, přehledné a jednoduché. Barvy byly voleny s ohledem na psychologii člověka tak, aby se se systémem jednoduše a příjemně pracovalo.

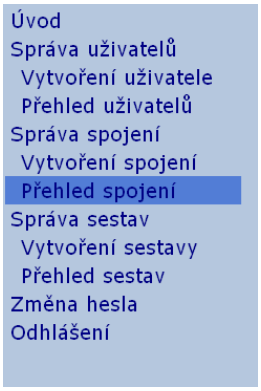
6.5.4 Hlavní okno aplikace



Obrázek 22: Hlavní okno aplikace

Hlavní okno aplikace je rozděleno na více částí. Skládá se z hlavičky, ve které je zobrazen název aplikace, dále z těla stránky, ve které je zobrazeno v levé části menu a v pravé části obsah stránky, který se mění v závislosti na zvolené položce v menu. Ve spodní části stránky je zobrazena patička se jménem autora. Hlavička i patička je zarovnána na střed obrazovky internetového prohlížeče.

6.5.5 Menu



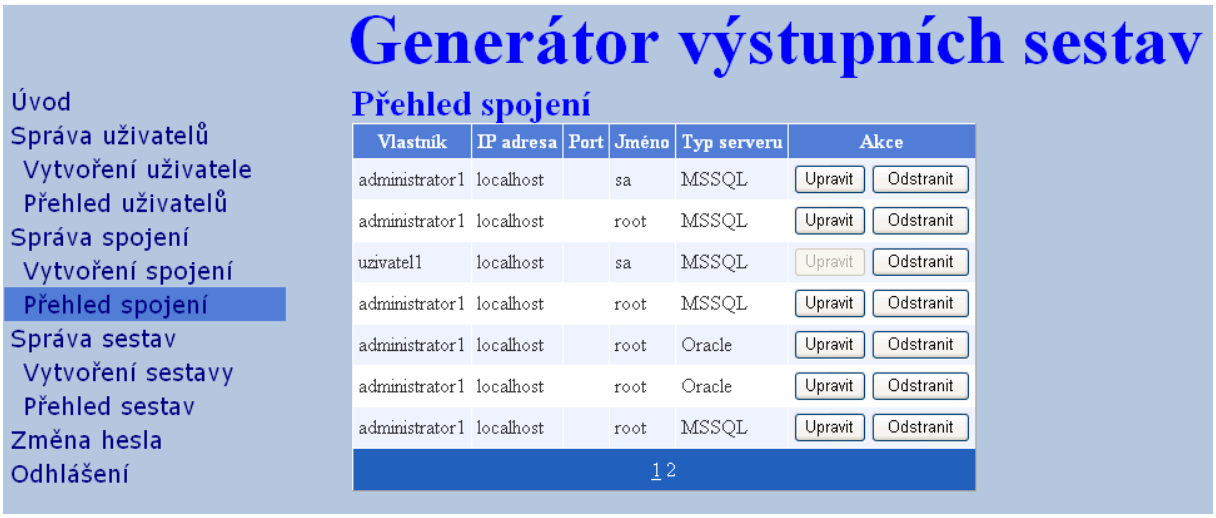
Jednotlivé funkce Generátoru výstupních sestav jsou přístupné prostřednictvím menu. Toto menu je víceúrovňové. V první úrovni jsou zobrazeny odkazy na obecné stránky v aplikaci jako úvodní stránka, stránka pro změnu hesla uživatele, stránka pro odhlášení ze systému a stránky pro správu jednotlivých položek systému. Odkazy pro správu vedou pouze na stránku pro výběr vytvoření položky nebo přehled položek. Odkazy na vytvoření a přehled vedou přímo na stránky věnované vytváření jednotlivých položek a stránky pro zobrazení položek systému. Zvýrazněná položka v menu označuje aktuálně zvolenou stránku aplikace.

Obrázek 23: Menu aplikace

6.5.6 Ukázka formuláře pro vytvoření nového spojení

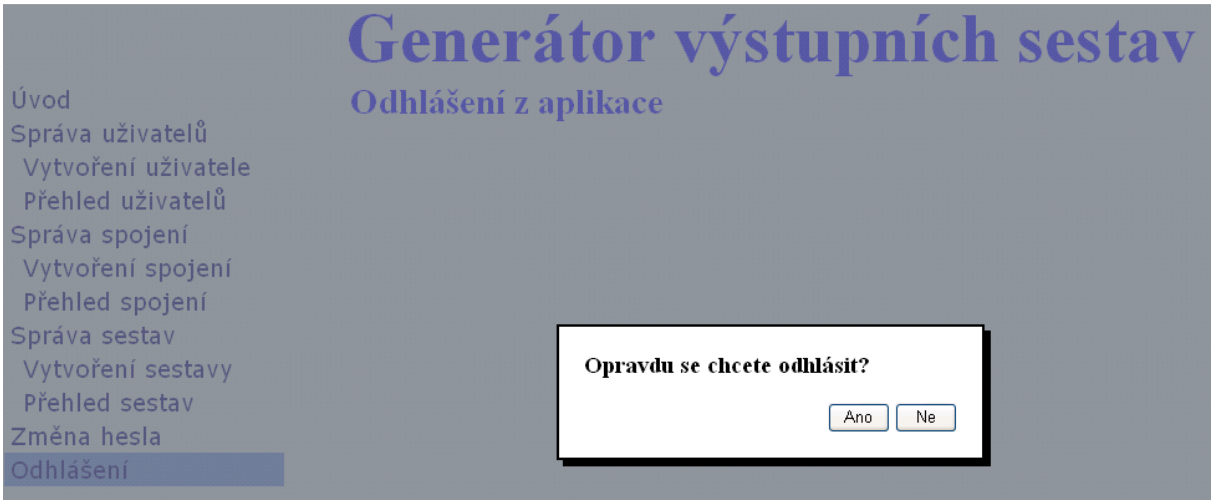
Obrázek 24: Formulář vytvoření nového spojení

6.5.7 Ukázka přehledu vytvořených spojení

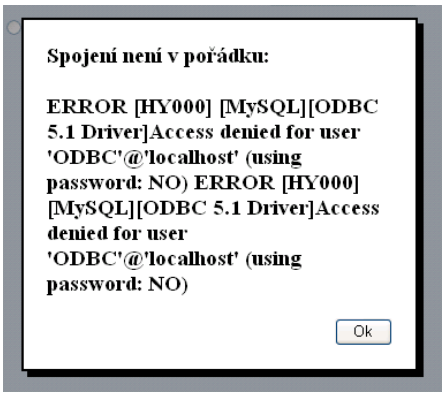


Obrázek 25: Přehled vytvořených spojení

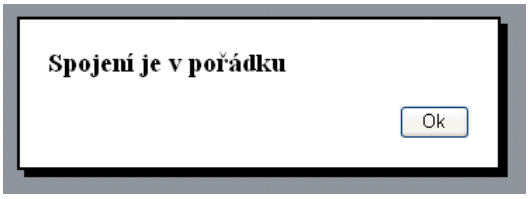
6.5.8 Ukázky dialogů pro zobrazení hlášení uživateli



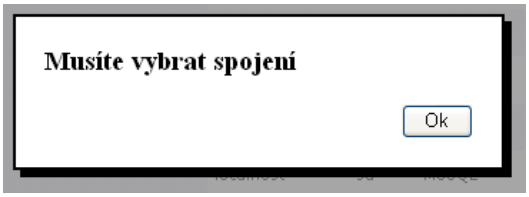
Obrázek 26: Dialog pro odhlášení ze systému



Obrázek 29: Chybový dialog při testu spojení



Obrázek 27: Dialog s informativním hlášením



Obrázek 28: Dialog s příkazovým hlášením

6.5.9 Ukázka jednoho kroku v průvodci pro vytvoření sestavy

Autor: Bc. Adam Bartoníček

Obrázek 30: Krok průvodce pro výběr atributů

6.6 Příručky k aplikaci

Příručky se nacházejí na přiloženém DVD v adresáři **/Prirucky**.

- **Uživatelská příručka**
  - Obsahuje popis ovládání aplikace pro uživatele.
- **Programátorská příručka**
  - Programátorská příručka se skládá z webových stránek. Příručka se spustí otevřením stránky *index.html*.
  - Obsahuje strukturu webových stránek a tříd, popis jednotlivých webových stránek a tříd a popis funkcí ve třídách.

6.7 Ukázka funkcí

Funkce níže jsou jen ukázkou funkcí použitých v aplikaci.

6.7.1 Funkce WizardStep5PridatAtributy

Tato funkce provede přidání označených atributů do výstupních atributů výsledné sestavy. Funkce nejprve zkontroluje, zda jsou vyplněny potřebné údaje, načte pole výstupních atributů a atributů příkazu z ViewState a poté prochází seznamem atributů zobrazeným v průvodci. Pokud je atribut označen, že se má v sestavě použít, tak se zkontroluje, zda již není v seznamu atributů. Pokud ano, použijí se jeho atributy a přidá se do seznamu výstupních atributů. Pokud není v seznamu atributů, tak se tam přidá a následně se také přidá do seznamu výstupních atributů. Nakonec se tyto seznamy uloží zpět do ViewState a provede se znovunačtení kroku průvodce.

6.7.1.1 Zdrojový kód funkce

```
protected void WizardStep5PridatAtributy(object sender, EventArgs e)
{
   // Pokud je uživatel přihlášený
   if (HttpContext.Current.User.Identity.IsAuthenticated)
   {
      if (GridView3.SelectedIndex >= 0)
      {
         string tabulka = GridView3.SelectedValue.ToString();

         if (GridView2.SelectedIndex >= 0)
         {

            prikazDet = (PrikazDetaily)ViewState["PrikazDet"];
            string databaze = GridView2.SelectedValue.ToString();

            databazeList = (ArrayList)ViewState["DatabazeList"];
            int dat = databazeList.BinarySearch(new DatabazeDetaily(null,
             databaze), new DatabazeDetailyPorovnatPodleNazvu());

            tabulkyList = (ArrayList)ViewState["TabulkyList"];
            int tab = tabulkyList.BinarySearch(new TabulkaDetaily(null,
             ((DatabazeDetaily)databazeList[dat]).id, tabulka), new
             TabulkaDetailyPorovnatPodleNazvu());

            atributyList = (ArrayList)ViewState["AtributyList"];

            prikazAtributyList =
             (SortedList)ViewState["PrikazAtributyList"];

            int pocitadlo = 0;
            if (prikazAtributyList.Count > 0)
            {
               pocitadlo =
               ((PrikazAtributDetaily)prikazAtributyList.GetKey(prikazAtrib
               utyList.Count - 1)).poradi + 1;
            }
            // Procházení seznamem atributů
```

```
            foreach (GridViewRow row in GridView6.Rows)
            {
               // Nalezení checkboxu
               CheckBox cb = (CheckBox)row.FindControl("Vybrat");
               TextBox tb = (TextBox)row.FindControl("Hlavicka");
               if (cb != null && cb.Checked)
               {

                  // Uložení hodnoty klíče pro daný řádek
                  string nazev =
                   GridView6.DataKeys[row.RowIndex].Value.ToString();
                  int atr = atributyList.BinarySearch(new
                   AtributDetaily(null,
                   ((TabulkaDetaily)tabulkyList[tab]).id, nazev), new
                   AtributDetailyPorovnatPodleNazvu());
                  AtributDetaily ad = new AtributDetaily();
                  if (atr >= 0)
                  {
                     ad.id = ((AtributDetaily)atributyList[tab]).id;
                     ad.idTabulka =
                       ((AtributDetaily)atributyList[tab]).idTabulka;
                     ad.nazev = ((AtributDetaily)atributyList[tab]).nazev;
                  }
                  else
                  {
                     ad.id = Guid.NewGuid().ToString();
                     ad.idTabulka = ((TabulkaDetaily)tabulkyList[tab]).id;
                     ad.nazev = nazev;
                     atributyList.Add(ad);
                  }

                  // Uložení hodnoty hlavičky pro daný atribut
                  string hlavicka = tb.Text;

                  PrikazAtributDetaily a = new
                   PrikazAtributDetaily(Guid.NewGuid().ToString(), nazev,
                   ad.id, prikazDet.id, null, hlavicka, pocitadlo, tabulka,
                   null);
                  prikazAtributyList.Add(a, a.id);
                  pocitadlo++;
               }
            }

            ViewState["PrikazAtributyList"] = prikazAtributyList;
            ViewState["AtributyList"] = atributyList;
            WizardStep5ZmenaStavuCheckbuttonu(false);
            PocatecniNastaveniKroku5();
            GridView4PrirazeniDat();
         }
      }
   }
}
```

6.7.2 Funkce StahnutiSkriptu

Funkce přesměruje požadavek na stránku, která jej zpracuje a nabídne ke stažení PHP skript.

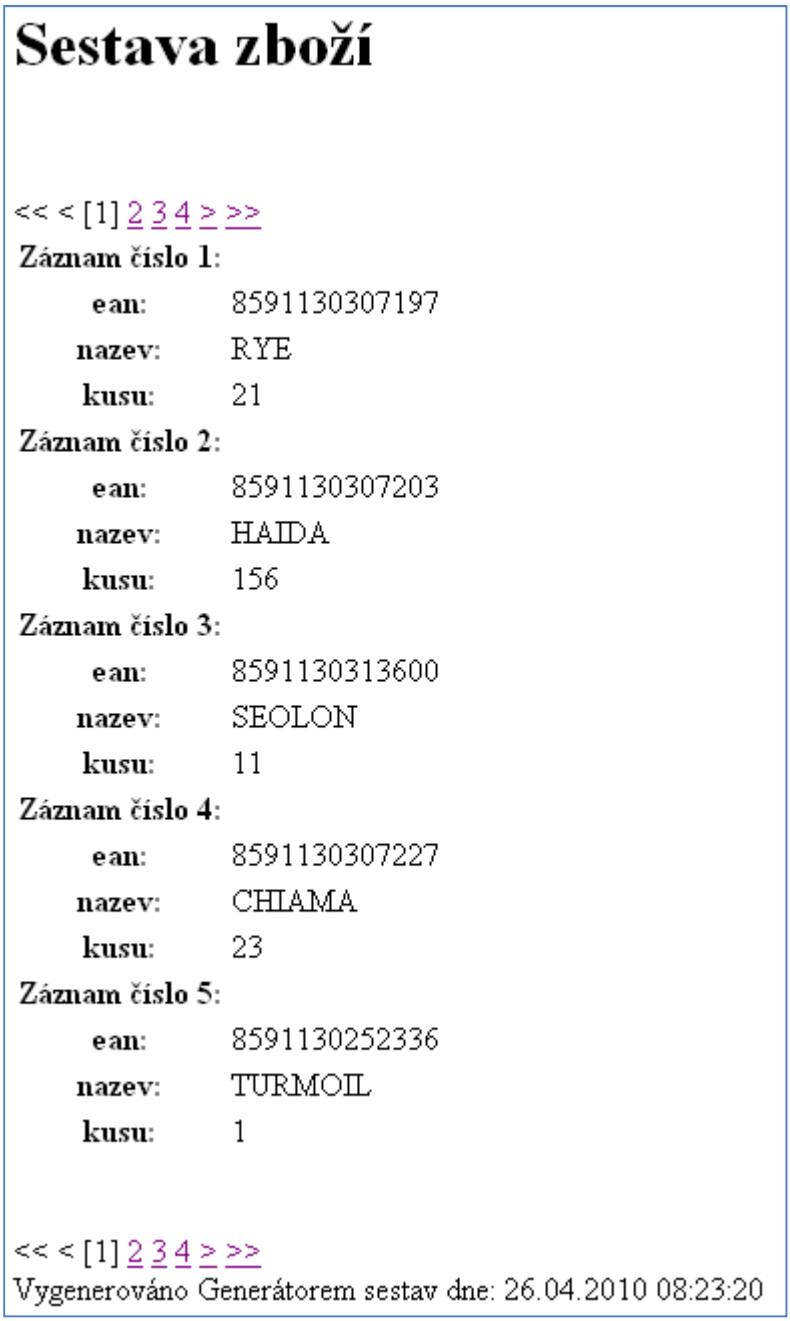
6.7.2.1 Zdrojový kód funkce

```
protected void StahnutiSkriptu(object sender, EventArgs e)
   {
      // Pokud je uživatel přihlášený
      if (HttpContext.Current.User.Identity.IsAuthenticated)
      {
        sestavaDet = (SestavaDetaily)ViewState["SestavaDet"];
        Response.Redirect(Server.HtmlEncode("~/Zabezpecene/SestavySablony/"
        + typSest.ZiskaniTypuSestavySablona(RadioButtonList1.SelectedValue)
        + ".ashx?id=" + sestavaDet.id));
      }
   }
```

6.8 Ukázka části PHP skriptu pro připojení k databázi

```
// Navázání spojení s databázovým serverem pomocí ODBC rozhraní
@$spojeni = odbc_connect("Driver={SQL Server Native Client
10.0};Server=localhost;Database=xxx;", "xxx", "xxx");
// Pokud spojení nebylo navázáno, výpis chybového hlášení a ukončení
skriptu
if(!$spojeni)
{
      // Výpis chybového hlášení
      echo "<div>Chyba při připojení k databázovému serveru přes roz-
hraní ODBC</div>";
      // Ukončení činnosti skriptu
      exit();
}
// Pokud spojení bylo navázáno
else
{
      // Zjištění čísla stránky, která se bude zobrazovat
      ...
      ...
      ...
```

6.9 Ukázka výstupu PHP skriptu



Sestava zboží

<< < [1] 2 3 4 > >>

Záznam číslo 1:

ean: 8591130307197
nazev: RYE
kusu: 21

Záznam číslo 2:

ean: 8591130307203
nazev: HAIDA
kusu: 156

Záznam číslo 3:

ean: 8591130313600
nazev: SEOLON
kusu: 11

Záznam číslo 4:

ean: 8591130307227
nazev: CHIAMA
kusu: 23

Záznam číslo 5:

ean: 8591130252336
nazev: TURMOIL
kusu: 1

<< < [1] 2 3 4 > >>
Vygenerováno Generátorem sestav dne: 26.04.2010 08:23:20

Obrázek 31: Ukázkový výstup PHP skriptu

7 Testování

Testování Generátoru sestav probíhalo pomocí testovacích scénářů. Některé z nich jsou uvedeny níže. Tyto testovací scénáře popisují manuální testování aplikace. Testovalo se pouze na testovacích datech, protože systém ještě není v plném provozu.

7.1 Testovací scénář Vytvoření nového spojení

Název scénáře:	Vytvoření nového spojení
Autor:	Bc. Adam Bartoníček
Datum vytvoření:	25.3.2010
Podmínky pro testování:	• Počítač s internetovým prohlížečem s funkčním připojením k internetu • spuštěný webový a databázový server • spuštěná aplikace Generátor sestav v internetovém prohlížeči • uživatel přihlášen v aplikaci • zobrazena stránka Vytvoření spojení
Vstupy:	Informace o konkrétním spojení
Očekávané výstupy:	Vložení nového záznamu do typu entity spojení a zobrazení potvrzovacího hlášení o úspěšném uložení spojení.
Skutečné výstupy:	Vložení nového záznamu do typu entity spojení a zobrazení potvrzovacího hlášení o úspěšném uložení spojení.

Tabulka 38: TS vytvoření nového spojení 1

Číslo	Popis	Výsledek
1	Kontrola, zda se zobrazil formulář se 4 textovými poli, 3 přepínacími tlačítky a dvěma tlačítky (Test, Uložit)	**OK**
2	Vyplnění formuláře testovacími údaji	**OK**
3	Kontrola funkce tlačítka Test	**OK**
4	Kontrola zobrazení hlášení o testu spojení	**OK**
5	Kontrola funkce tlačítka Uložit	**OK**
6	Kontrola uložení hodnot z formuláře do paměťových proměnných	**OK**
7	Kontrola, zda se do typu entity Spojeni vložil nový záznam s údaji z paměťových proměnných	**OK**
8	Kontrola zobrazení hlášení o uložení spojení	**OK**

Tabulka 39: TS vytvoření nového spojení 2

Výsledek testu: Test proběhl v pořádku.
Poznámky:

7.2 Testovací scénář Zobrazení prvků na stránce

Název scénáře:	Zobrazení prvků na stránce
Autor:	Bc. Adam Bartoníček
Datum vytvoření:	25.3.2010
Podmínky pro testování:	• Počítač s internetovým prohlížečem s funkčním připojením k internetu • spuštěný webový a databázový server
Vstupy:	Uživatelovi přihlašovací informace
Očekávané výstupy:	Shodné a správné zobrazení testovaných stránek ve všech prohlížečích
Skutečné výstupy:	Zobrazení v prohlížečích se mírně lišilo

Tabulka 40: TS Zobrazení prvků na stránce 1

Číslo	Popis	Výsledek
1	Spuštění internetového prohlížeče	**OK**
2	Zadání URL adresy Generátoru sestav	**OK**
3	Kontrola, zda se zobrazila přihlašovací stránka	**OK**
4	Kontrola zobrazení prvků na přihlašovací stránce	**CHYBA**
5	Zadání přihlašovacích údajů uživatele	**OK**
6	Kontrola, zda po stisku tlačítka Přihlásit došlo k přihlášení do aplikace	**OK**
7	Kontrola, zda se zobrazilo menu aplikace	**CHYBA**
8	Kontrola, zda se zobrazila úvodní stránka aplikace	**CHYBA**
9	Kontrola funkce menu zvolením stránka pro přehled uživatelů	**OK**
10	Kontrola, zda se zobrazila opravdu stránka s přehledem uživatelů	**OK**
11	Kontrola zobrazení prvků na této stránce	**CHYBA**
12	Odhlášení z aplikace pomocí volby Odhlášení v menu aplikace	**OK**
13	Kontrola správného zobrazení dialogového okna na zašeděném pozadí aplikace.	**OK**
14	Zadání volby Ano	**OK**
15	Kontrola, zda se zobrazila přihlašovací stránka	**OK**

Tabulka 41: TS Zobrazení prvků na stránce 2

Výsledek testu: Test proběhl s chybami. Chyby byly způsobeny rozdílným zobrazením v různých prohlížečích. Chyby se týkaly použití různých fontů písma v různých prohlížečích, jiných mezer mezi ovládacími prvky, odlišnými barvami okrajů tabulek. Tyto chyby ovšem nemají vliv na funkčnost. Každý prohlížeč zobrazil všechny ovládací prvky na stránce a vždy na správném místě. Umístění se lišilo v prohlížečích jen o velmi málo pixelů. Všechny ovládací prvky na stránce plnily svou funkci. Tyto chyby se nebudou odstraňovat, nemají vliv na funkčnost a vzhled se liší pouze minimálně.
Poznámky: Tento test bude probíhat postupně na více internetových prohlížečích.
Těmito prohlížeči budou:

Internet Explorer 8, Mozilla Firefox 3.6, Opera 10.5, Apple Safari 4.0.4

7.3 Testovací scénář Úprava existujícího uživatele

Název scénáře:	Úprava existujícího uživatele
Autor:	Bc. Adam Bartoníček
Datum vytvoření:	25.3.2010
Podmínky pro testování:	• Počítač s internetovým prohlížečem s funkčním připojením k internetu • spuštěný webový a databázový server • spuštěná aplikace Generátor sestav v internetovém prohlížeči • uživatel přihlášen v aplikaci • zobrazena stránka Přehled uživatelů • Existence uživatele **aaa** v databázi s atributy (aaa; uživatel; aaa@aaa.cz; 25.3.2010 20:31:54; 25.3.2010 20:31:54; 0; 1;1) – tento uživatel byl vytvořen jen pro potřeby testování
Vstupy:	Informace o konkrétním uživateli
Očekávané výstupy:	Úprava uživatele v databázi
Skutečné výstupy:	Úprava uživatele v databázi

Tabulka 42: TS úprava existujícího uživatele 1

Číslo	Popis	Výsledek
1	Kontrola, zda je v přehledu zobrazen uživatel **aaa**	**OK**
2	Výběr tohoto uživatele pomocí tlačítka Upravit na řádku s informacemi o něm	**OK**
3	Kontrola zobrazení formuláře pro úpravu uživatele	**CHYBA**
4	Změna role uživatele z uživatel na Administrátor	**OK**
5	Stisknutí tlačítka Uložit	**OK**
6	Kontrola, zda se tato změna projevila v paměťových proměnných	**OK**
7	Kontrola, zda se tato změna provedla i v databázi	**OK**
8	Kontrola, zda se zobrazilo hlášení o změně uživatele	**OK**

Tabulka 43: TS úprava existujícího uživatele 2

Výsledek testu: Test proběhl s chybami. Nezobrazila se volba, zda je uživatel schválený pro přihlášení, či nikoliv. Tento problém byl ihned po zjištění chyby odstraněn změnou programového kódu. Po příštím spuštění testu se tato chyba již neprojevila.

Poznámky:

8 Závěr

V této práci jsem vytvořil kompletní analýzu Generátoru výstupních sestav. Na základě této analýzy jsem implementoval systém, který poskytuje možnost generovat výstupní sestavy z databází. Finální podoba systému odpovídá zadání s tím, že nejsou implementovány všechny typy výstupních sestav, ale do systému naopak přibyly funkce navíc, které vyplynuly se zadání. Výsledná aplikace byla testována ve více prostředích a na reálných datech. Na testování systému se podíleli lidé, kteří neměli s realizací systému přímé vazby. Tito lidé byli jak laikové, tak i zkušení testeři systémů.

Na aplikaci je ještě co zlepšovat. Průvodce pro vytváření výstupních sestav by jistě šel udělat přehlednější a s více možnostmi. Také by bylo možné přidat do generování PHP skriptu šablony pro vzhled pomocí CSS stylů, aby bylo možné výstupní sestavu přizpůsobit požadavkům uživatelů. Dále by bylo možné upravit průvodce tak, aby vytváření sestavy probíhalo graficky pomocí přetahování jednotlivých prvků na plátno, které by představovalo výslednou výstupní sestavu. Toto je již ale budoucí rozšiřování funkcionality.

8.1 Teoretický závěr

I když jsem měl jisté znalosti o databázích a vytváření výstupních sestav z nich, po konzultacích se zadavatelem jsem získal širší pohled na problém. Bylo tedy nutné, abych se důkladněji seznámil s problematikou generování výstupních sestav z databází a tomu přizpůsobit celý návrh aplikace. Musel jsem tudíž prostudovat odbornou literaturu, abych si rozšířil své dosavadní znalosti, a teprve poté jsem mohl začít navrhovat výsledný systém.

Vzhledem k mým malým zkušenostem s vývojem webových aplikací v ASP.NET jsem musel nastudovat velké množství literatury, abych pochopil specifika návrhu webových aplikací.

8.2 Praktický závěr

Tato práce popisuje vývoj aplikace od zadání až po praktickou realizaci. Aplikaci nešlo implementovat okamžitě. Celý projekt musel projít postupným vývojem od zadání, přes analýzu až po implementaci, aby jej bylo možné realizovat v příslušném rozsahu a vymezeném časovém horizontu. Vzhledem k drobným chybám při první analýze a implementaci jsem musel provést opakování tohoto vývojového cyklu, aby výsledná aplikace odpovídala požadavkům.

Při implementaci jsem narazil na problémy odlišnosti tvorby výstupních sestav z různých databázových serverů. Musel jsem prostudovat dokumentace k těmto serverům, abych mohl implementovat komunikaci s nimi. Díky tomuto jsem získal zkušenosti s prací s databázovými servery, které pro mě byly do té doby neznámé. Dále jsem získal spoustu zkušeností s vývojem webových aplikací v ASP.NET, protože jsem narážel na problémy, se kterými jsem se dosud nesetkal a musel jsem je vyřešit. V tomto mi hodně pomohla diskusní fóra, kde lidé řešili podobné problémy jako já.

Seznam literatury

1. *.NET Framework – Wikipedia, the free encyclopedia.* [Online] 15. Únor 2010. [Citace: 15. 02 2010.] <http://en.wikipedia.org/wiki/.NET_Framework>.

2. Martin, Timm. *What is .NET?* [Online] 2007. [Citace: 15. 02 2010.] <http://www.devtopics.com/what-is-net/>.

3. *ASP.NET – Wikipedia, the free encyclopedia.* [Online] 15. 2 2010. [Citace: 15. Únor 2010.] <http://en.wikipedia.org/wiki/ASP.NET>.

4. *PHP – Wikipedie, otevřená encyklopedie.* [Online] 6. Únor 2010. [Citace: 15. Únor 2010.] <http://cs.wikipedia.org/wiki/Php>.

5. *PHP - Wikipedia, the free encyclopedia.* [Online] 13. Únor 2010. [Citace: 15. Únor 2010.] <http://en.wikipedia.org/wiki/PHP>.

6. Group, The PHP. *PHP: Hypertext Preprocessor.* [Online] 15. Únor 2010. [Citace: 15. Únor 2010.] <http://php.net/index.php>.

7. Muknšnábl, Ing. Josef. *ODBC v kostce > r e b o o t . c z | it magazin | hackers - coders - admins - geeks & girls.* [Online] 30. Duben 2001. [Citace: 15. Únor 2010.] <http://reboot.cz/howto/databaze/odbc-v-kostce/articles.html?id=152>.

8. *Open Database Connectivity - Wikipedia, the free encyclopedia.* [Online] 13. Únor 2010. [Citace: 15. Únor 2010.] <http://en.wikipedia.org/wiki/Open_Database_Connectivity>.

9. *MySQL - Wikipedie, otevřená encyklopedie.* [Online] 5. Únor 2010. [Citace: 15. Únor 2010.] <http://cs.wikipedia.org/wiki/MySQL>.

10. *MySQL - Wikipedia, the free encyclopedia.* [Online] 15. Únor 2010. [Citace: 15. Únor 2010.] <http://en.wikipedia.org/wiki/MySQL>.

11. *MySQL :: The World's most popular open source database.* [Online] 2010. [Citace: 15. Únor 2010.] <http://www.mysql.com/>.

12. *Microsoft SQL Server - Wikipedia, the free encyclopedia.* [Online] 14. Únor 2010. [Citace: 15. Únor 2010.] <http://en.wikipedia.org/wiki/Microsoft_SQL_Server>.

13. Microsoft. *Microsoft SQL Server 2008.* [Online] 2009. [Citace: 15. Únor 2010.] <http://www.microsoft.com/cze/sqlserver2008/default.mspx>.

14. *Oracle Database - Wikipedia, the free encyclopedia.* [Online] 15. Únor 2010. [Citace: 15. Únor 2010.] <http://en.wikipedia.org/wiki/Oracle_Database>.

15. Oracle. *Oracle 11g, Siegel, PeopleSoft.* [Online] [Citace: 15. Únor 2010.] <http://www.oracle.com/index.html>.

16. Šarmanová, Jana. *Teorie zpracování dat.* 1. vydání. Ostrava : VŠB - TUO, 2003. str. 74. Sv. 1.

17. Šarmanová, Jana. *Databázové a informační systémy.* 1. vydání. Ostrava : VŠB - TUO, 2007. str. 122. Sv. 1. ISBN 978-80-248-1499-5.

18. Šarmanová, Jana. *Informační systémy a datové sklady.* 1. vydání. Ostrava : VŠB - TUO, 2007. str. 169. Sv. 1. ISBN 978-80-248-1500-8.

19. Lacko, Luboslav. *SQL Hotová řešení.* Brno : Computer press, 2003. str. 298. ISBN: 80-7226-975-5.

20. MacDonald, Matthew a Szpuszta, Mario. *ASP.NET 3.5 a C# 2008 tvorba dynamických stránek profesionálně.* Brno : ZONER press, 2008. str. 1584. ISBN 978-80-7413-008-3.

21. Microsoft. Forums.asp.net - Home. [Online] 2010. [Citace: 22. Duben 2010.] <http://forums.asp.net/>.

22. —. MSDN Library. [Online] 2010. [Citace: 22. duben 2010.] <http://msdn.microsoft.com/en-us/library/>.

23. —. *Hardware and Software Requirements for Installing SQL Server 2008.* [Online] 31. Srpen 2009. [Citace: 23. Březen 2010.] http://msdn.microsoft.com/en-us/library/ms143506.aspx.

Seznam příloh

- **Příloha č. 1**: E-R diagram s popisem atributů pro datovou analýzu
- **Příloha č. 2**: Datová analýza
- **Příloha č. 3**: Návrh implementace
- **Příloha č. 4**: E-R diagram s popisem atributů pro návrh implementace
- **Příloha č. 5**: Uživatelská příručka
- **Příloha č. 6**: Funkční analýza

Adresářová struktura přiloženého DVD

/Aplikace	Verze aplikace, která jde spustit (nainstalovat) na webovém serveru
/Prilohy	Adresář obsahuje přílohy k práci
/Prirucky	Uživatelská a programátorská příručka k aplikaci
/Programatorska	Programátorská příručka k aplikaci
/Uzivatelska	Uživatelská příručka k aplikaci
/Programy	Programy potřebné pro instalaci aplikace
/Apache	Adresář obsahuje instalátor webového serveru Apache
/MSSQL	Adresář obsahuje instalátor SQL serveru 2008 a nástrojů
/MySQL	Adresář obsahuje instalátor MySQL serveru a nástrojů
/NETframework	Adresář obsahuje instalátor .NET Frameworku 3.5 + SP1
/ODBC	Adresář obsahuje instalátory pro ODBC ovladače jednotlivých serverů
/Oracle	Adresář obsahuje instalátor pro Oracle server a nástroje
/PHP	Adresář obsahuje instalátor jazyka PHP
/Sql	Skript pro vytvoření databáze a její naplnění daty
/Src	Adresář obsahuje zdrojový kód aplikace
/Projects/GeneratorSestav	Adresář obsahuje projekt do Visual Studia 2008
/WebSites/GeneratorSestav	Adresář obsahuje zdrojový kód projektu
/Text	Adresář obsahuje textovou část práce